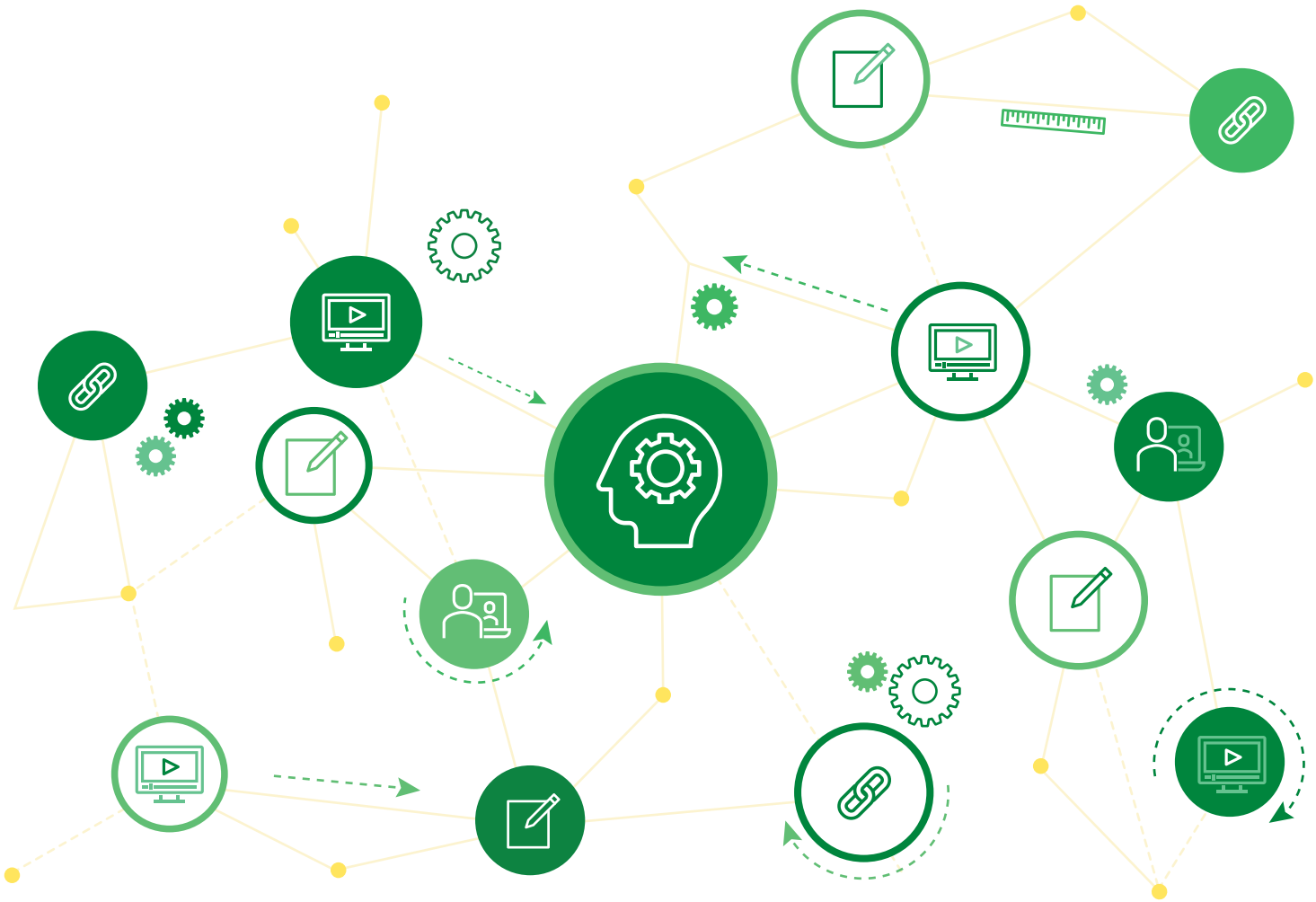
T³ FRANCE

Teachers Teaching with Technology™



Laurent Didier et Florent Girod

Livret d'activités en SNT
Lycée général

www.t3france.fr

python™

Sommaire

Compétences visées

Un des objectifs de l'enseignement de SNT est de développer et de coder des scripts PYTHON afin d'apporter une réponse à une problématique précise. A travers le thème "**informatique embarquée et objets connectés**", nous pouvons notamment travailler les compétences suivantes dans l'activité proposée :

- Coder des scripts simples d'acquisition de données.
- Gérer des entrées/sorties à travers les ports utilisés par le système.
- Écrire et développer des algorithmes pour résoudre une problématique.
- Identifier des algorithmes de contrôle des comportements physiques à travers les données des capteurs.

Situation déclenchante

Comment allumer une lampe (sa lampe de chevet par exemple) sans appuyer parfaitement sur l'interrupteur ? (si, au matin, on n'est pas très bien réveillé !)

Ici, une **vidéo** illustrant ce projet.

Problématique

Comment reproduire un allumage à distance d'une lampe ?

1. Identifier un capteur possible ;
2. Identifier un actionneur possible ;
3. Élaborer un algorithme modélisant le fonctionnement d'une lampe qui s'allume à distance ;
4. Mettre cet algorithme en application.

Matériel disponible

Les élèves disposent, en plus de leur TI-83 Premium CE :

- d'un TI-Innovator HUB ;
- d'un capteur de distance à ultrasons.

Déroulement possible du projet

En amont du projet :

Les propositions suivantes permettent de préparer le projet et de le rendre possible sur 2 séances d'1h30. Elles peuvent aussi être préparées par l'enseignant sous forme de fiches et/ou d'un exposé au tableau.

- Un groupe d'élèves peut préparer une fiche ou présenter un rapide exposé illustrant les principaux points pour maîtriser la programmation en PYTHON spécifique au **TI-Innovator HUB**, en s'appuyant sur les **'10 mn de code'** (Unité 6).
- Un groupe d'élèves peut expliquer le **mode de codage RGB** pour les couleurs.
- Un groupe d'élèves peut décrire le **TI-Innovator HUB**.

Différentes évolutions possibles pour le projet :

Les propositions suivantes donnent des pistes pour gérer les différences de vitesse d'exécution des élèves.
On peut aussi considérer ces différentes étapes comme une manière de séquencer le projet pour ceux qui en auraient besoin.

- 1ère étape : on allume une lumière en passant sa main devant le capteur à ultrasons.
- 2ème étape : on allume et on éteint une lumière en passant sa main devant le capteur à ultrasons ; création d'un compteur et discrimination selon sa parité.
- 3ème étape : on adapte l'intensité lumineuse de la lampe selon la luminosité ambiante.

A la suite du projet :

Synthétiser ce travail par rapport au cours de SNT :

- le(s) capteur(s) et le(s) actionneur(s) mis en jeu ;
- l'algorithme qui gère les données d'entrée ;
- le langage de programmation utilisé.

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Proposition de résolutions

Choix du capteur :

- Capteur de distance à ultrasons : en passant sa main à proximité, la distance mesurée passe en dessous d'un seuil qui entraîne le passage d'un mode à l'autre. C'est ce capteur qui a été choisi ici.
- Capteur de luminosité (brightness) : en cachant ce capteur, la luminosité passe en dessous d'un seuil qui entraîne le passage d'un mode à l'autre. On peut l'utiliser si on ne dispose pas de capteur à ultrasons ou si on veut proposer une alternative.

Choix de l'actionneur :

- LED RGB du HUB : elle pourra délivrer une lumière blanche plus ou moins intense.

Etapes de résolution

L'importation de ces bibliothèques est explicitée dans la remarque qui suit cette partie.

```
EDITEUR : LAMPED
LIGNE DU SCRIPT 0007
# Projets STEM Hub
from ti_system import *
from time import *

import color
from ranger import *
Fns… | a A # | Outils | Exéc | Script
```

La fonction **f1** va simuler un allumage automatique de la lumière à distance :

On initialise un compteur stocké dans la variable **c** qui comptera le nombre de fois où l'on passe la main à moins de 10 cm du capteur de distance.
La variable **n** donnera la parité de **c** (par **n=c%2**, reste de la division euclidienne de **c** par 2).
Ces deux variables sont initialisées à 0.

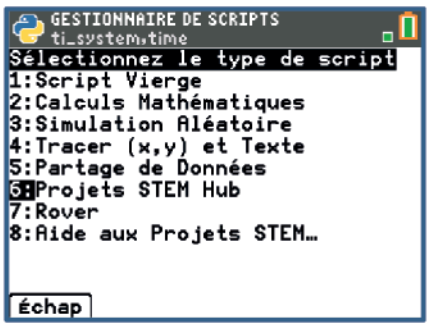
On crée la variable **m** qui représente l'entrée du ranger (capteur de distance) relié au port d'entrée **IN 1** ; les instructions se trouvent dans le module *ranger* importé précédemment.

```
ÉDITEUR : LAMPED
LIGNE DU SCRIPT 0018
def f1():
    c=0
    n=0
    m=ranger("IN 1")
    while not escape():
        d=m.measurement()
        sleep(0.5)
        if d<0.1:
            c+=1
        n=c%2
        if n==0:
            color.rgb(0,0,0)
        else:
            color.rgb(255,255,255)
Fns… | a A # | Outils | Exéc | Script
```

- `while not escape():` crée une boucle 'infinie' qui sera interrompue par l'appui sur la touche **on** ; l'instruction se trouve dans le module *ti_system*.
- `d=m.measurement()` crée la variable **d** qui est la mesure donnée par **m** (le ranger branché à l'entrée 1).
- Une mesure est réalisée chaque demi-seconde.
- Si la distance mesurée par le ranger est inférieure à 0,1 m = 10 cm (en passant la main à proximité), le compteur **c** s'incrémente d'une unité.
- rappel : `c+=1` est un raccourci pour `c=c+1`
- Si **n** est pair, on éteint la led en saisissant la fonction **color.rgb(0,0,0)** ; la fonction **color.rgb** se trouve dans la bibliothèque *color* importée en préambule.
- Sinon, on saisit **color.rgb(255,255,255)**, ce qui a pour effet de mettre la led en lumière blanche à son maximum d'intensité.

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Remarque



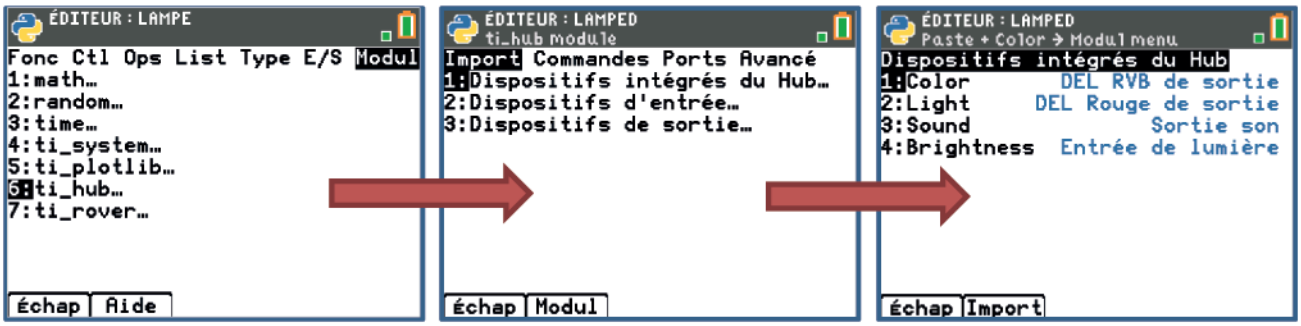
En préambule du code, on trouve l'importation de bibliothèques spécifiques au projet :

- elles peuvent être importées une à une en allant les chercher dans le menu **Modul** ;
- elles peuvent s'implémenter en choisissant le type de programme « **Projets STEM Hub** » (choix 6 des types de programmes proposés) à la création du nouveau programme.

La bibliothèque *ti_system* permet d'importer ou d'exporter des données dans les listes de la calculatrice.
Pour ce projet, elle contient l'instruction **while not escape( ) :** qui permet de lancer une boucle qui ne s'interrompt que lorsqu'on appuie sur la touche **on**.

La bibliothèque *time* permet en particulier d'utiliser la fonction **sleep** qui prend comme argument un temps de pause donné en secondes.

On va ensuite importer une bibliothèque spécifique au projet, à savoir la bibliothèque *color* en suivant les instructions suivantes :



Au fur et à mesure que les bibliothèques sont importées, elles s'ajoutent aux bibliothèques déjà présentes (*math*, *random*, …etc.) et permettent d'accéder aux instructions qu'elles contiennent.

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Pour aller plus loin

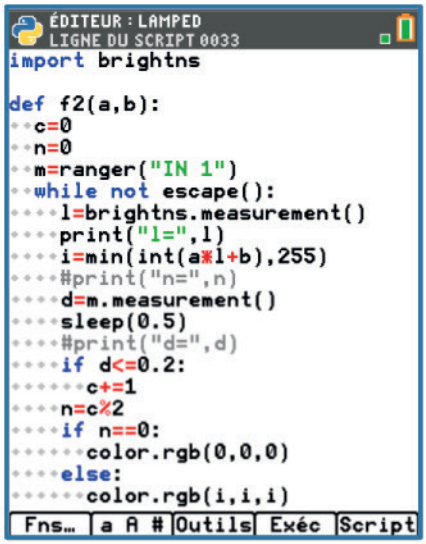
On peut demander **une intensité lumineuse dépendant de la luminosité ambiante**. Pour cela, on mesurera la luminosité par le capteur de luminosité intégré au HUB et on transformera la valeur captée en un nombre entier compris entre 0 et 255.

C'est l'objet de la capture d'écran ci-contre qui montre la fonction **f2**.

Elle a pour paramètres **a** et **b** qui seront à choisir par l'utilisateur selon la luminosité ambiante et l'effet voulu :

- Très peu de lumière si l'ambiance est sombre ;
- Ou, au contraire, une luminosité d'autant plus grande que l'ambiance est sombre.

Une fonction affine est à déterminer selon le choix des élèves et la luminosité ambiante qu'il faudra mesurer pour avoir une idée des valeurs données qui varient d'un jour à l'autre.



Remarques

Quelques remarques sur cet approfondissement possible :

- Le module **brigthns** a été importé comme décrit précédemment comme un des dispositifs intégrés au HUB.

  Une fois intégré, il apparaît parmi les autres modules et permet de charger l'instruction **brighns.measurement**.

- On peut demander quelques affichages pour aider au bon réglage des paramètres.

  C'est par exemple ce que permet **print("l = ",l)** qui affichera la luminosité ambiante.

  Si cet affichage devient inutile, on peut le commenter en précédant l'instruction de # (qui s'obtient notamment par le raccourci "touche **2nde** suivie de la touche **3**") ; c'est ce qui a été fait dans le programme proposé.

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Mélange de couleurs

F. GIROD

Compétences visées

Un des objectifs de l'enseignement de SNT est de développer et de coder des scripts PYTHON afin d'apporter une réponse à une problématique précise. A travers le thème "**informatique embarquée et objets connectés**", nous pouvons notamment travailler les compétences suivantes dans l'activité proposée :

- Coder des scripts simples d'acquisition de données.
- Gérer des entrées/sorties à travers les ports utilisés par le système.
- Écrire et développer des algorithmes pour résoudre une problématique.
- Identifier des algorithmes de contrôle des comportements physiques à travers les données des capteurs.

Situation déclenchante

Comment créer et modifier facilement des couleurs en utilisant le code RGB ?

Ici, une **vidéo** illustrant ce projet.

Problématique

Comment modifier un code RGB facilement et en voir les effets immédiats ?

1. Identifier un capteur possible ;
2. Identifier un (des) actionneur(s) possible(s) ;
3. Élaborer un algorithme modélisant la création de couleurs en code RGB (au moins deux couleurs parmi les trois) ;
4. Mettre cet algorithme en application.

Matériel disponible

Les élèves disposent, en plus de leur TI-83 Premium CE :

- d'un TI-Innovator HUB ;
- de deux capteurs de distance à ultrasons.

Déroulement possible du projet

En amont du projet :

Les propositions suivantes permettent de préparer le projet et de le rendre possible sur 2 séances d'1h30. Elles peuvent aussi être préparées par l'enseignant sous forme de fiches et/ou d'un exposé au tableau.

- Un groupe d'élèves peut préparer une fiche ou présenter un rapide exposé illustrant les principaux points pour maîtriser la programmation en PYTHON spécifique au **TI-Innovator HUB** en s'appuyant sur les **'10 mn de code'** (Unité 6).
- Un groupe d'élèves peut expliquer le **mode de codage RGB** pour les couleurs.
- Un groupe d'élèves peut décrire le **TI-Innovator HUB**.

Différentes évolutions possibles pour le projet :

Les propositions suivantes donnent des pistes pour gérer les différences de vitesse d'exécution des élèves.
On peut aussi considérer ces différentes étapes comme une manière de séquencer le projet pour ceux qui en auraient besoin.

- 1ère étape : on mesure la distance entre sa main et un premier capteur à ultrasons ; même chose avec un second capteur.
- 2ème étape : on utilise ces données en les transformant en un nombre entier compris entre 0 et 255.
- 3ème étape : on produit une image RG ou RB ou GB avec les deux valeurs précédemment trouvées.

A la suite du projet :

Synthétiser ce travail par rapport au cours de SNT : le(s) capteur(s) et le(s) actionneur(s) mis en jeu / l'algorithme qui gère les données d'entrée / le langage de programmation utilisé.

Proposition de résolutions

Choix du capteur :

- Capteur de distance à ultrasons : en passant sa main à proximité, la distance mesurée nous permet de modifier la valeur de R, G ou B.

  Remarque : on ne peut utiliser que deux capteurs car seules les entrées IN 1 et IN 2 permettent de brancher un tel capteur.

Choix des actionneurs :

- LED RGB du HUB : elle pourra délivrer une lumière de la couleur de son choix ; c'est cet actionneur qui est choisi ici.

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Etapes de résolution

L'importation de ces bibliothèques est explicitée dans la remarque qui suit cette partie.

```
ÉDITEUR : COULEURS
LIGNE DU SCRIPT 0011
# Projets STEM Hub
from ti_system import *
from time import *

import color
from ranger import *
Fns… | a A # | Outils | Exéc | Script
```

r1=ranger("IN 1") crée la variable **r1** qui est de type 'ranger'.
r1 et **r2** représentent les entrées des deux rangers (capteurs de distance) reliés aux ports d'entrée **IN 1** et **IN 2**.
Les instructions se trouvent dans le module *ranger* importé précédemment.

```
ÉDITEUR : COULEURS
LIGNE DU SCRIPT 0001
# Projets STEM Hub
from ti_system import *
from time import *

import color
from ranger import *

r1=ranger("IN 1")
r2=ranger("IN 2")
Fns… | a A # | Outils | Exéc | Script
```

- `while not escape():` crée une boucle 'infinie' qui sera interrompue par l'appui sur la touche **on** ; l'instruction se trouve dans le module *ti_system*.
- `m1=r1.measurement()` crée la variable **m1** qui stocke la mesure donnée par **r1** (le ranger branché à l'entrée 1).
- La variable **r** permet de transformer la valeur **m1** en un nombre entier compris entre 0 et 255 ; de même pour **b** avec la valeur de **m2**.
- L'instruction **color.rgb(r,0,b)** permet de colorer la led avec les paramètres **r** et **b**. Cette fonction se trouve dans la bibliothèque *color* importée en préambule.
  - En mettant un obstacle près du ranger branché en **IN 1**, on rend la valeur proche **r** de 0 : la led est proche d'un bleu pur.
  - En mettant un obstacle près du ranger branché en **IN 2**, on rend la valeur proche **b** de 0 : la led est proche d'un rouge pur.
  - En mettant un obstacle à une distance équivalente des deux rangers, on obtient des violets.

```
ÉDITEUR : COULEURS
LIGNE DU SCRIPT 0022
while not escape():
··m1=r1.measurement()
··m2=r2.measurement()
··#print("m1=",m1)
··#print("m2=",m2)
··r=min(255,int(255/0.5*m1))
··b=min(255,int(255/0.5*m2))
··#print("r=",r)
··#print("b=",b)
··color.rgb(r,0,b)
··#sleep(1)
Fns… | a A # | Outils | Exéc | Script
```

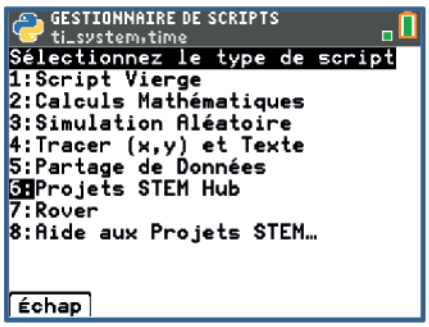
- Pour faciliter la compréhension du script en cours de réalisation, on peut proposer des affichages.
  - Affichage des valeurs de **m1**, **m2**, **r** et **b**. Si les valeurs de **r** et **b** ne conviennent pas, on peut modifier à sa guise les lignes de calculs correspondantes.
  - Pour que l'affichage ait le temps d'être lu, on peut proposer une pause par l'instruction **sleep(1)** (pause d'une seconde) ; cette instruction se trouve dans la bibliothèque *time*.
  - Pour commenter ces instructions par la suite, on insère un **#** (par exemple par le raccourci : touche **2nde** suivi de la touche **3**).

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Remarques

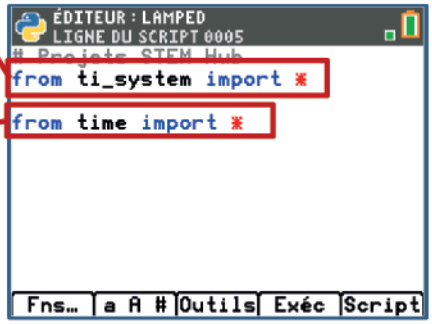
En préambule du code, on trouve l'importation des bibliothèques spécifiques au projet :

- elles peuvent être importées une à une en allant les chercher dans le menu **Modul** ;
- elles peuvent s'implémenter en choisissant le type de programme « **Projets STEM Hub** » (choix 6 des types de programmes proposés) à la création du nouveau programme.

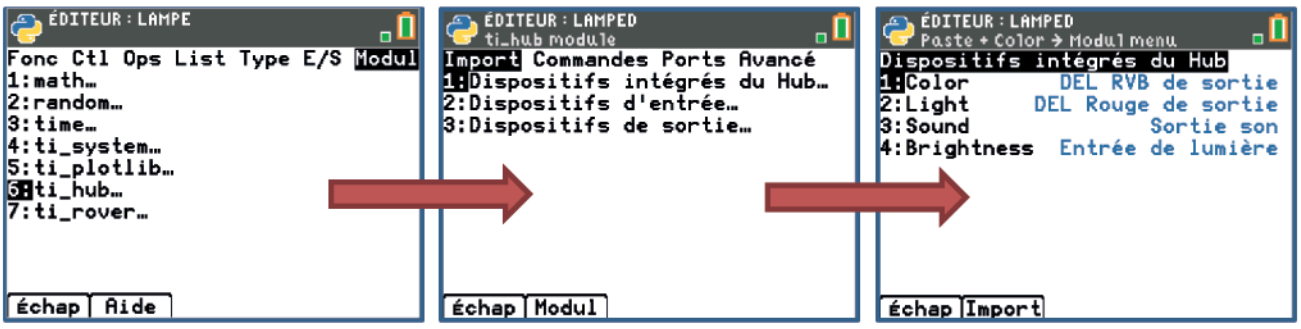


La bibliothèque *ti_system* permet d'importer ou d'exporter des données dans les listes de la calculatrice.
Pour ce projet, elle contient l'instruction **while not escape() :** qui permet de lancer une boucle qui ne s'interrompt que si on appuie sur la touche **on**.

La bibliothèque *time* permet en particulier d'utiliser la fonction **sleep** qui prend comme argument un temps de pause donné en secondes.



On va ensuite importer une bibliothèque spécifique au projet, à savoir la bibliothèque *color* en suivant les instructions suivantes :



Au fur et à mesure que les bibliothèques sont importées, elles s'ajoutent aux bibliothèques déjà présentes (*math*, *random*, …etc.) et permettent d'accéder aux instructions qu'elles contiennent.
On procède de même pour importer la bibliothèque *ranger* qui s'obtiendra en choisissant '**dispositifs d'entrée**' à l'étape 2 de la procédure précédente.

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Lampe frontale

F. GIROD

Compétences visées

Un des objectifs de l'enseignement de SNT est de développer et de coder des scripts PYTHON afin d'apporter une réponse à une problématique précise. A travers le thème "**informatique embarquée et objets connectés**", nous pouvons notamment travailler les compétences suivantes dans l'activité proposée :

- Coder des scripts simples d'acquisition de données.
- Gérer des entrées/sorties à travers les ports utilisés par le système.
- Écrire et développer des algorithmes pour résoudre une problématique.
- Identifier des algorithmes de contrôle des comportements physiques à travers les données des capteurs.

Situation déclenchante

Une lampe frontale propose en général plusieurs modes :

- éteinte ;
- lumière faible ;
- lumière plus soutenue ;
- lumière rouge ;
- lumière rouge clignotante.

Comment reproduire un fonctionnement de ce type ?

Ici, une vidéo illustrant le projet.

Problématique

Comment reproduire le fonctionnement d'une lampe frontale ?

1. Identifier un capteur possible ;
2. Identifier un actionneur possible ;
3. Élaborer un algorithme modélisant le fonctionnement d'une lampe frontale ;
4. Mettre cet algorithme en application.

Matériel disponible

Les élèves disposent, en plus de leur TI-83 Premium CE :

- d'un TI-Innovator HUB ;
- d'un capteur de distance à ultrasons.

Fiche méthode

Déroulement possible du projet

En amont du projet :

Les propositions suivantes permettent de préparer le projet et de le rendre possible sur 2 séances d'1h30. Elles peuvent aussi être préparées par l'enseignant sous forme de fiches et/ou d'un exposé au tableau.

- Un groupe d'élèves peut préparer une fiche ou présenter un rapide exposé illustrant les principaux points pour maîtriser la programmation en PYTHON spécifique au **TI-Innovator HUB**, en s'appuyant sur les **'10 mn de code'** (Unité 6)
- Un groupe d'élèves peut expliquer le **mode de codage RGB** pour les couleurs.
- Un groupe d'élèves peut décrire le **TI-Innovator HUB**.

Différentes évolutions possibles pour le projet :

Les propositions suivantes donnent des pistes pour gérer les différences de vitesse d'exécution des élèves.
On peut aussi considérer ces différentes étapes comme une manière de séquencer le projet pour ceux qui en auraient besoin.

- 1ère étape : on allume une lumière en passant sa main devant le capteur à ultrasons.
- 2ème étape : on allume et on éteint une lumière en passant sa main devant le capteur à ultrasons : création d'un compteur et discrimination selon sa parité.
- 3ème étape : on actionne plusieurs modes (au choix des élèves):
    - On passe sa main devant le capteur : une lumière peu intense s'allume ;
    - On passe à nouveau sa main : une lumière plus intense s'allume ;
    - On passe à nouveau sa main : une lumière rouge s'allume ;
    - On passe à nouveau sa main : la lumière s'éteint ;
    - On passe à nouveau sa main : le cycle recommence.
- 4ème étape : on met en place un mode "clignotement".

A la suite du projet :

Synthétiser ce travail par rapport au cours de SNT :

- le(s) capteur(s) et le(s) actionneur(s) mis en jeu ;
- l'algorithme qui gère les données d'entrée ;
- le langage de programmation utilisé.

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Proposition de résolutions

Choix du capteur :

- Capteur de distance à ultrasons : en passant sa main à proximité, la distance mesurée passe en dessous d'un seuil qui entraîne le passage d'un mode à l'autre. C'est ce capteur qui a été choisi ici.
- Capteur de luminosité (brightness) : en cachant ce capteur, la luminosité passe en dessous d'un seuil qui entraîne le passage d'un mode à l'autre. On peut l'utiliser si on ne dispose pas de capteur à ultrasons ou si on veut proposer une alternative.

Choix de l'actionneur :

- LED RGB du HUB : elle pourra délivrer une lumière blanche plus ou moins intense ou une lumière colorée.

Etapes de résolution

L'importation de ces bibliothèques est explicitée dans la remarque qui suit cette partie.

```
ÉDITEUR : LAMPEF
LIGNE DU SCRIPT 0001
# Projets STEM Hub
from ti_system import *
from time import *

import color
from ranger import *

c=0
n=0

m=ranger("IN 1")
Fns… | a A # | Outils | Exéc | Script
```

On crée les variables **c** et **n** qui seront des compteurs. Elles sont initialisées à 0.

- **c** comptera le nombre de fois où l'on passe près du capteur de distance.
- **n** donnera le reste de la division euclidienne de **c** par 5 (cela s'écrit en python **c%5**) : cela permet d'activer une des cinq modalités de la lampe.

La variable **m** représente l'entrée du ranger (capteur de distance) relié au port d'entrée **IN 1** ; les instructions se trouvent dans le module **ranger** importé précédemment.

```
ÉDITEUR : LAMPEF
LIGNE DU SCRIPT 0025
while not escape():
  print("n=",n)
  d=m.measurement()
  sleep(0.5)
  print("d=",d)
  if d<0.1:
    c+=1
  n=c%5
  if n==0:
    color.rgb(0,0,0)
  elif n==1:
    color.rgb(100,100,100)
  elif n==2:
    color.rgb(255,255,255)
  elif n==3:
    color.rgb(255,0,0)
  else:
    color.rgb(255,0,0)
    sleep(0.1)
    color.rgb(0,0,0)
    sleep(0.1)
Fns… | a A # | Outils | Exéc | Script
```

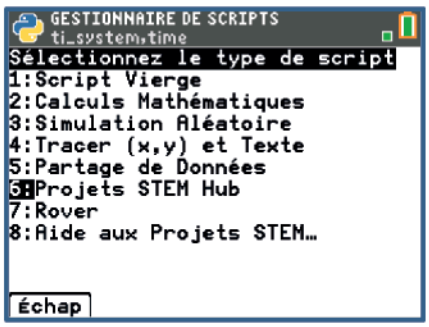
- `while not escape():` crée une boucle 'infinie' qui sera interrompue par l'appui sur la touche **on** ; l'instruction se trouve dans le module *ti_system*.
- `d=m.measurement()` crée la variable **d** qui est la mesure donnée par **m** (le ranger branché à l'entrée 1).
- Une mesure est réalisée chaque demi-seconde.
- Si la distance mesurée par le ranger est inférieure à 0,1 m = 10 cm (en passant la main à proximité), le compteur **c** s'incrémente d'une unité.
- Rappel : **c+=1** est un raccourci pour **c=c+1**
- Selon la valeur de **n** (0/1/2/3/4), on aura diverses situations :
  - lumière éteinte avec **color.rgb(0,0,0)** (cette fonction se trouve dans la bibliothèque *color* importée en préambule) ;
  - lumière blanche peu lumineuse avec **color.rgb(100,100,100)** ;
  - lumière blanche lumineuse avec **color.rgb(255,255,255)** ;
  - lumière rouge avec **color.rgb(255,0,0)** ;
  - lumière rouge clignotante (temps de clignotement de 0,1 s).

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Remarques

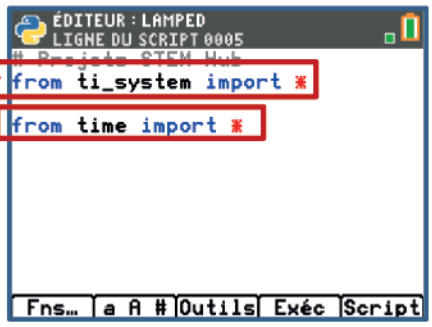
En préambule du code, on trouve l'importation de bibliothèques spécifiques au projet :

- elles peuvent être importées une à une en allant les chercher dans le menu **Modul** ;
- elles peuvent s'implémenter en choisissant le type de programme « **Projets STEM Hub** » (choix 6 des types de programmes proposés) à la création du nouveau programme.

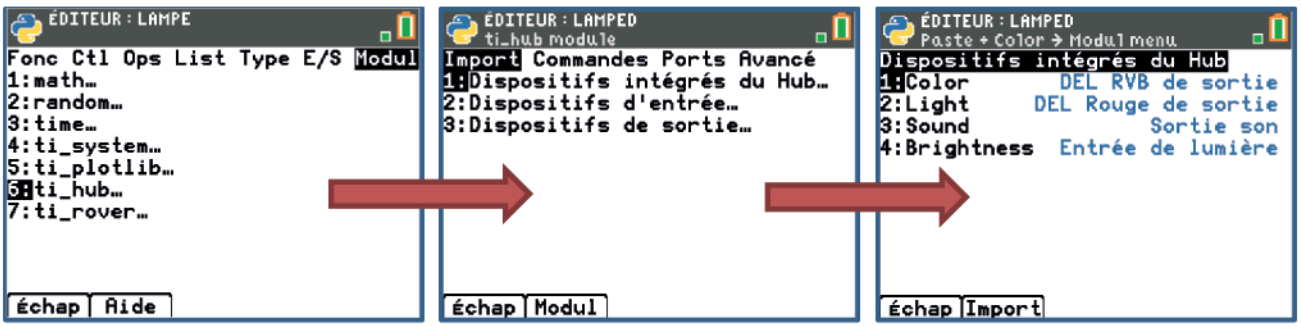


La bibliothèque *ti_system* permet d'importer ou d'exporter des données dans les listes de la calculatrice.
Pour ce projet, elle contient l'instruction **while not escape( ) :** qui permet de lancer une boucle qui ne s'interrompt que lorsqu'on appuie sur la touche **on**.

La bibliothèque *time* permet en particulier d'utiliser la fonction **sleep** qui prend comme argument un temps de pause donné en secondes.



On va ensuite importer une bibliothèque spécifique au projet, à savoir la bibliothèque *color* en suivant les instructions suivantes :



Au fur et à mesure que les bibliothèques sont importées, elles s'ajoutent aux bibliothèques déjà présentes (*math*, *random*, …etc.) et permettent d'accéder aux instructions qu'elles contiennent.

On procède de même pour importer la bibliothèque *Ranger* qui s'obtiendra en choisissant '**dispositifs d'entrée**' à l'étape 2 de la procédure précédente.

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Radar de recul

F. GIROD

Compétences visées

Un des objectifs de l'enseignement de SNT est d'écrire et développer des programmes pour répondre à des problèmes concrets. A travers le thème "**informatique embarquée et objets connectés**", nous pouvons notamment travailler les compétences suivantes dans l'activité proposée :

- Écrire des programmes simples d'acquisition de données.
- Gérer des entrées/sorties à travers les ports utilisés par le système.
- Écrire et développer des programmes pour répondre à des problèmes.
- Identifier des algorithmes de contrôle des comportements physiques à travers les données des capteurs.

Situation déclenchante

Comment signaler au conducteur d'une voiture que son véhicule approche d'un obstacle lorsqu'il recule ?

Ici, une **vidéo** illustrant le projet.

Problématique

Comment reproduire le fonctionnement d'un radar de recul ?

1. Identifier un capteur possible ;
2. Identifier un actionneur ou des actionneurs possibles ;
3. Mettre en place un algorithme modélisant le fonctionnement d'un radar de recul ;
4. Le mettre en application.

Matériel disponible

Les élèves disposent, en plus de leur TI-83 Premium CE :

- d'un TI-Innovator HUB ;
- d'un capteur de distance à ultrasons.

Fiche méthode

Déroulement possible du projet

En amont du projet :

Les propositions suivantes permettent de préparer le projet et de le rendre possible sur 2 séances d'1h30. Elles peuvent aussi être préparées par l'enseignant sous forme de fiche et/ou d'un exposé au tableau.

- Un groupe d'élèves peut préparer une fiche ou présenter un rapide exposé illustrant les principaux points pour maîtriser la programmation en python spécifique au **TI-Innovator HUB** en s'appuyant sur les **'10 mn de code'** (Unité 6)
- Un groupe d'élèves peut expliquer le **mode de codage RGB** pour les couleurs.
- Un groupe d'élèves peut décrire le **TI-Innovator HUB**.

Différentes évolutions possibles pour le projet :

Les propositions suivantes donnent des pistes pour gérer les différences de vitesse d'exécution des élèves.
On peut aussi considérer ces différentes étapes comme une manière de séquencer le projet pour ceux qui en auraient besoin.

- 1ère étape : on émet un bip (à une certaine fréquence) lorsque la distance détectée par le capteur est inférieure à une certaine valeur.
- 2ème étape : on émet différents types de 'bip' selon la distance à l'obstacle.
- 3ème étape : on émet un bip dont la fréquence dépend continument de la distance à l'obstacle.
- 4ème étape : on double ce bip par un clignotement lumineux.

A la suite du projet :

Synthétiser ce travail par rapport au cours de SNT :

le/les capteurs et le/les actionneurs mis en jeu / l'algorithme qui gère les données d'entrée / le langage de programmation utilisé.

Proposition de résolutions

Choix du capteur :

- Capteur de distance à ultrasons : en passant sa main à proximité, la distance mesurée passe en dessous d'un seuil qui entraîne l'émission d'un bip ou/et d'un clignotement lumineux.

Choix de / des actionneurs :

- LED RGB du HUB : elle pourra délivrer une lumière de la couleur de son choix.
- LED du HUB : elle émet une lumière rouge : il est choisi dans ce projet.
- Haut-parleur du HUB : il émet un son dont on définit la fréquence d'émission et la durée : il est choisi dans ce projet.

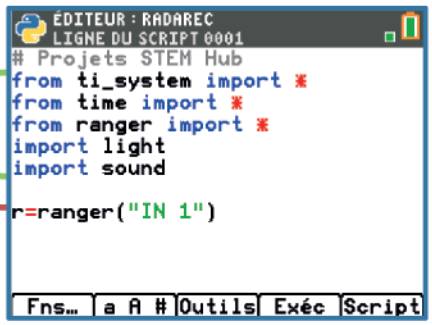
Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Fiche méthode

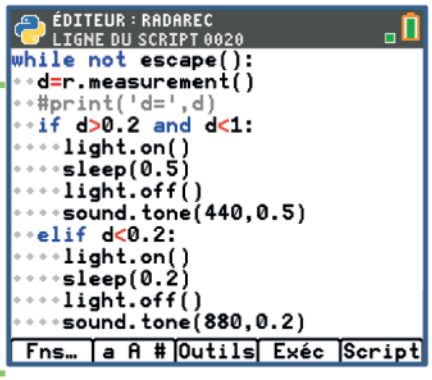
Etapes de résolution

L'importation de ces bibliothèques est explicitée dans la remarque qui suit cette partie.

On crée la variable **r** qui représente l'entrée du ranger (capteur de distance) relié au port d'entrée **IN 1** ; les instructions se trouve dans le module **ranger** importé précédemment.



- `while not escape():` crée une boucle 'infinie' qui sera interrompue par l'appui sur la touche **on** ; l'instruction se trouve dans le module **ti_system**
- `d=m.measurement()` crée la variable **d** qui est la mesure donnée par **r** (le ranger branché à l'entrée 1)
- On propose un affichage de **d** (pendant la réalisation du programme, pour s'assurer que tout fonctionne correctement), affichage que l'on peut commenter par la suite par la suite par # (qui s'obtient par exemple par la touche **2nde** suivie de la touche **3**)
- selon la valeur de **d** , on aura plusieurs cas de figure :
  - distance comprise entre 20 cm et 1 m : on fait clignoter la lumière rouge du hub (toutes les 0,5 seconde) et on émet un bip (son de 440 Hz toutes les 0,5 seconde)
  - distance inférieure à 20 cm : on fait clignoter la lumière rouge du hub (toutes les 0,2 seconde) et on émet un bip plus aigu (son de 880 Hz toutes les 0,5 seconde)
  - distance supérieure à 1 m : il ne se passe rien ; pas besoin d'écrire d'instructions.
- La fonction **sleep** se trouve dans la bibliothèque **time**.
- La fonction **sound.tone** se trouve dans la bibliothèque **sound**.
- Les fonctions **light.on** et **light.off** se trouvent dans la bibliothèque light.



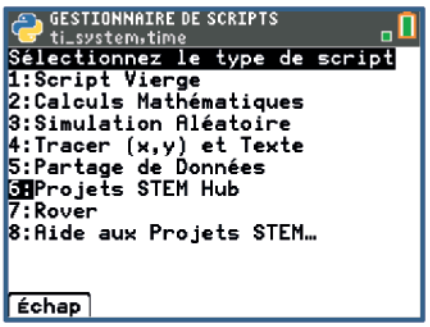
Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Fiche méthode

Remarques

En préambule du code, importation de bibliothèques spécifiques au projet :

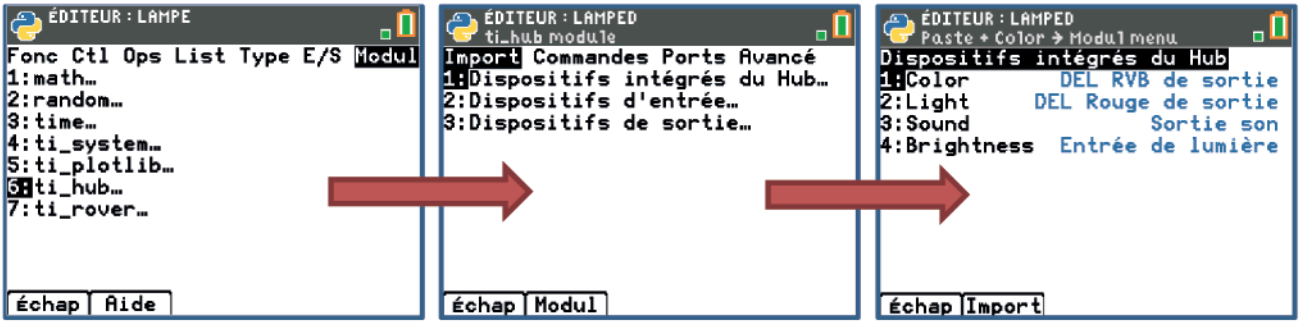
- Elles peuvent être importées une à une en allant les chercher dans le menu **Modul** ;
- Elles peuvent s'implémenter en choisissant le type de programme « **Projets STEM Hub** » (choix 6 des types de programmes proposés) à la création du nouveau programme.



La bibliothèque « **ti_system** » permet d'importer ou d'exporter des données dans les listes de la calculatrice.
Pour ce projet, elle contient l'instruction **while not escape() :** qui permet de lancer une boucle qui ne s'interrompt que si on appuie sur la touche **on**.

La bibliothèque « **time** » permet en particulier d'utiliser la fonction **sleep** qui prend comme argument un temps de pause donné en secondes.

On va ensuite importer une bibliothèque spécifique au projet, à savoir la bibliothèque « **light** » en suivant les instructions suivantes :



Au fur et à mesure que les bibliothèques sont importées, elles s'ajoutent aux bibliothèques déjà présentes (**math**, **random**, etc.) et permettent d'accéder aux instructions qu'elles contiennent.

On procède de même pour importer la bibliothèque **sound** ; la bibliothèque **ranger** s'obtiendra en choisissant '**dispositifs d'entrée**' à l'étape 2 de la procédure précédente.

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Fiche méthode

Pour aller plus loin

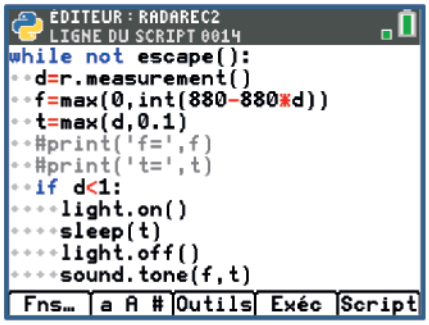
On propose sur la base de ce projet l'approfondissement suivant qui donne un rendu plus réaliste pour le radar de recul :

un 'bip' qui devient de plus en plus fréquent et de plus en plus aigu au fur et à mesure que l'on se rapproche de l'obstacle

L'objectif est double :

- Complexifier le projet.
- Le rendre plus proche de la réalité.

Pour cela, on peut créer deux variables : **f** et **t** qui dépendent de la variable **d** :



- On a choisi pour cette fréquence un maximum de 880 Hz avec un son qui commence à être audible à moins de 1 m.
  Il faut s'assurer que la valeur de **f** est positive, c'est pourquoi on a utilisé la fonction **max** ; par ailleurs, la valeur de **f** doit être un nombre entier, d'où l'utilisation de **int**.

- On a choisi un intervalle de temps entre deux bips égal à **d**.
  Il faut s'assurer que cette valeur soit au minimum égale à 0,1 pour que la fonction **sound.tone** fonctionne. C'est pourquoi on a utilisé la fonction **max** avec la valeur 0.1 pour être sûr de ne pas passer sous cet intervalle de temps.

On pourra bien sûr modifier les expressions de **f** et **t**. Il faudra être attentif à ce que :

- Les valeurs soient du bon format (entier pour **f**).
- Les valeurs soient dans des intervalles compatibles avec l'effet voulu (des fréquences trop aigues ou trop graves ne sont pas audibles).
- Les valeurs aient des valeurs compatibles avec le fonctionnement des fonctions associées (un temps inférieur à 0,1 s pour **sound.tone** ne convient pas).

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Radar pédagogique

F. GIROD

Compétences visées

Un des objectifs de l'enseignement de SNT est de développer et de coder des scripts PYTHON afin d'apporter une réponse à une problématique précise. A travers le thème "**informatique embarquée et objets connectés**", nous pouvons notamment travailler les compétences suivantes dans l'activité proposée :

- Coder des scripts simples d'acquisition de données.
- Gérer des entrées/sorties à travers les ports utilisés par le système.
- Écrire et développer des algorithmes pour résoudre une problématique.
- Identifier des algorithmes de contrôle des comportements physiques à travers les données des capteurs.

Situation déclenchante

Comment signaler à un conducteur d'un véhicule qu'il dépasse la vitesse autorisée ?

Ici, une **vidéo** illustrant le projet.

Problématique

Comment reproduire le fonctionnement d'un radar pédagogique ?

1. Identifier un capteur possible ;
2. Identifier un actionneur ou des actionneurs possibles ;
3. Élaborer un algorithme modélisant la mesure d'une vitesse ;
4. Mettre cet algorithme en application.

Matériel disponible

Les élèves disposent, en plus de leur TI-83 Premium CE :

- d'un TI-Innovator HUB ;
- d'un capteur de distance à ultrasons.

Déroulement possible du projet

En amont du projet :

Les propositions suivantes permettent de préparer le projet et de le rendre possible sur 2 séances d'1h30. Elles peuvent aussi être préparées par l'enseignant sous forme de fiches et/ou d'un exposé au tableau.

- Un groupe d'élèves peut préparer une fiche ou présenter un rapide exposé illustrant les principaux points pour maîtriser la programmation en python spécifique au **TI-Innovator HUB** en s'appuyant sur les **'10 mn de code'** (Unité 6).
- Un groupe d'élèves peut expliquer le **mode de codage RGB** pour les couleurs.
- Un groupe d'élèves peut décrire le **TI-Innovator HUB**.

Différentes évolutions possibles pour le projet :

Les propositions suivantes donnent des pistes pour gérer les différences de vitesse d'exécution des élèves.
On peut aussi considérer ces différentes étapes comme une manière de séquencer le projet pour ceux qui en auraient besoin.

- 1ère étape : on mesure la distance entre le véhicule et le capteur toutes les t secondes.
- 2ème étape : on utilise ces données pour déterminer la distance parcourue pendant ce laps de temps.
- 3ème étape : on calcule la vitesse moyenne du véhicule durant ce laps de temps.
- 4ème étape : on signale par un code couleur que la vitesse est acceptable ou non.

A la suite du projet :

Synthétiser ce travail par rapport au cours de SNT : le(s) capteur(s) et le(s) actionneur(s) mis en jeu / l'algorithme qui gère les données d'entrée / le langage de programmation utilisé.

Proposition de résolutions

Choix du capteur :

- Capteur de distance à ultrasons : on mesure la distance entre le capteur et le véhicule entre deux temps donnés, ce qui permet de calculer la vitesse moyenne sur cette durée.

Choix de / des actionneurs :

- LED RGB du HUB : elle pourra délivrer une lumière de la couleur de son choix : c'est cet actionneur qui est choisi ici ; rouge si la vitesse est trop importante, vert si la vitesse est conforme.
- Haut-parleur du HUB : il émet un son dont on définit la fréquence d'émission et la durée ; on choisit ici d'émettre un bip si la vitesse autorisée est dépassée.

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Etapes de résolution

L'importation de ces bibliothèques est explicitée dans la remarque qui suit cette partie.

La variable **r** représente l'entrée du ranger (capteur de distance) relié au port d'entrée **IN 1** ; les instructions se trouvent dans le module *Ranger* importé précédemment.
On crée les variables **t** et **va** qui pourront être modifiées par l'utilisateur.

- **t** est la durée entre deux mesures de distance ;
- **va** est la limite de vitesse que l'on s'autorise.

```
EDITEUR : RADARP
LIGNE DU SCRIPT 0001
# Projets STEM Hub
from ti_system import *
from time import *
from ranger import *
import color
import sound

r=ranger("IN 1")
t=0.1
va=5
```

Fns… | a A # | Outils | Exéc | Script

- `while not escape():` crée une boucle 'infinie' qui sera interrompue par l'appui sur la touche **on** ; l'instruction se trouve dans le module *ti_system*.
- `d=m.measurement()` crée la variable **d** qui est la mesure donnée par **m** (le ranger branché à l'entrée 1).
  - On effectue deux mesures : c'est pourquoi on a créé deux variables **d1** et **d2**.
  - Un intervalle de temps **t** sépare ces deux mesures. **sleep** se trouve dans la bibliothèque *time*.
- **v** est la vitesse moyenne sur la durée **t** :
  - On utilise la fonction **abs** (valeur absolue) pour pouvoir donner la vitesse aussi bien quand on se rapproche que quand on s'éloigne du capteur de distance.
  - la vitesse est en $km.h^{-1}$ (du fait de la multiplication par 3,6)
- Selon la valeur de **v**, on aura deux cas de figure :
  - vitesse autorisée : on allume une lumière verte par **color.rgb(0,255,0)** (la fonction **color.rgb** se trouve dans la bibliothèque *color* importée en préambule).
  - vitesse dépassée : on allume une lumière rouge par **color.rgb(255,0,0)** et on émet un bip aigu (fréquence de 880 Hz toutes les 0,1 seconde).

  A noter que la fonction **sound.tone** se situe dans la bibliothèque *sound* importée en préambule et que 0,1 est la durée minimale pour laquelle cette fonction est opérationnelle.

```
EDITEUR : RADARP
LIGNE DU SCRIPT 0022
while not escape():
  d1=r.measurement()
  sleep(t)
  d2=r.measurement()
  v=abs(d2-d1)/t*3.6
  #print("V = ",v,"km/h")
  if v<va:
    color.rgb(0,255,0)
  else:
    color.rgb(255,0,0)
    sound.tone(880,0.1)
    sleep(t)
```

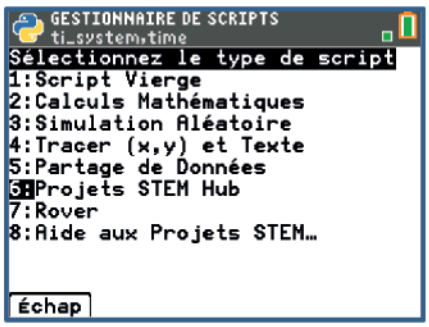
Fns… | a A # | Outils | Exéc | Script

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Remarques

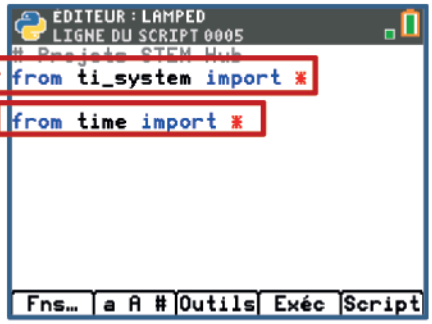
En préambule du code, on trouve l'importation de bibliothèques spécifiques au projet :

- Elles peuvent être importées une à une en allant les chercher dans le menu **Modul** ;
- Elles peuvent s'implémenter en choisissant le type de programme « **Projets STEM Hub** » (choix 6 des types de programmes proposés) à la création du nouveau programme.

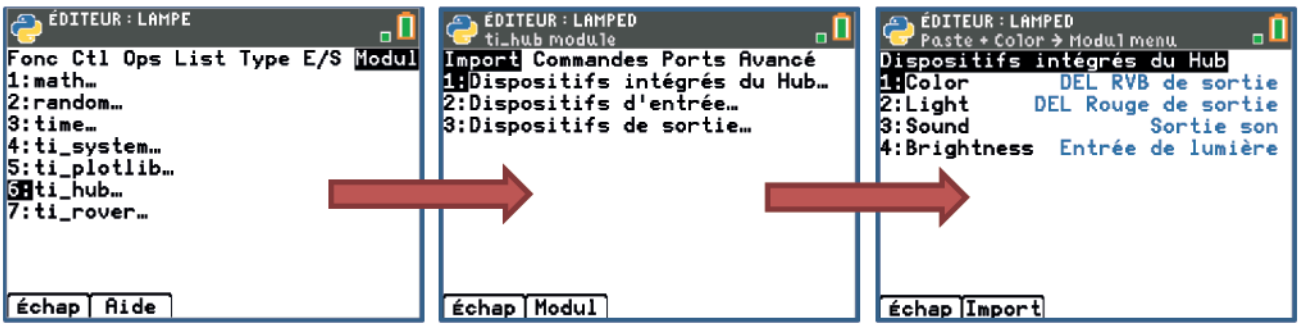


La bibliothèque *ti_system* permet d'importer ou d'exporter des données dans les listes de la calculatrice.
Pour ce projet, elle contient l'instruction **while not escape( ) :** qui permet de lancer une boucle qui ne s'interrompt que si on appuie sur la touche **on**.

La bibliothèque *time* permet en particulier d'utiliser la fonction **sleep** qui prend comme argument un temps de pause donné en secondes.



On va ensuite importer une bibliothèque spécifique au projet, à savoir la bibliothèque *color* en suivant les instructions suivantes :



Au fur et à mesure que les bibliothèques sont importées, elles s'ajoutent aux bibliothèques déjà présentes (*math*, *random,* etc.) et permettent d'accéder aux instructions qu'elles contiennent.

On procède de même pour importer la bibliothèque *Ranger* qui s'obtiendra en choisissant '**dispositifs d'entrée**' à l'étape 2 de la procédure précédente.

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Thème : informatique embarquée et objets connectés

Classe enseignée : SNT

L.DIDIER

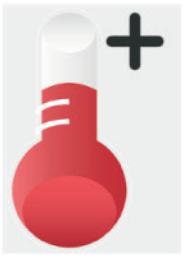
Ventilateur autonome

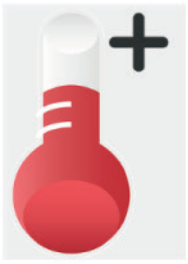
Compétences visées

Un des objectifs de l'enseignement de SNT est de développer et de coder des scripts PYTHON afin d'apporter une réponse à une problématique précise. A travers le thème " **informatique embarquée et objets connectés**", nous pouvons notamment travailler les compétences suivantes dans l'activité proposée :

- Coder des scripts simples d'acquisition de données.
- Gérer des entrées/sorties à travers les ports utilisés par le système.
- Écrire et développer des algorithmes pour résoudre une problématique.
- Identifier des algorithmes de contrôle des comportements physiques à travers les données des capteurs.

Situation déclenchante





Aujourd'hui, la technologie permet de contrôler des objets à travers les données de capteurs. Par exemple l'éclairage ou la régulation de la température d'une pièce peuvent se faire de manière automatique.

Problématique

Comment concevoir une solution technologique pour déclencher le fonctionnement d'un ventilateur ou l'ouverture d'une fenêtre si la température d'une pièce est trop élevée en présence d'une personne ?

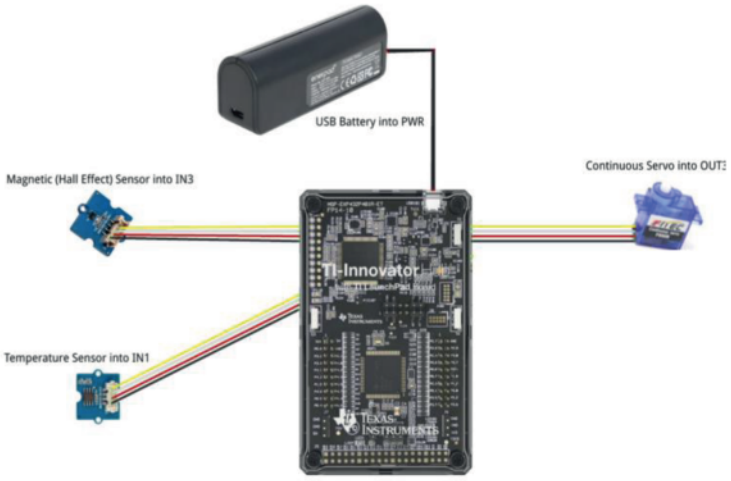
Thème : informatique embarquée et objets connectés

Classe enseignée : SNT

Fiche méthode

L.DIDIER

Matériel nécessaire



- Calculatrice TI-83 Premium CE
- Câble (calculatrice/Hub)
- TI-Innovator Hub
- Câble (3)
- Capteur de température
- Batterie externe
- Servomoteur
- Câble de batterie

Déroulement possible du projet

Travail de groupe possible. Chaque groupe disposera du matériel ci-dessus et devra concevoir une réponse à la problématique sous forme d'une maquette munie d'une documentation qui explique les scripts pilotant la maquette. A la fin du projet, les groupes pourront voter pour élire la production qui répond le mieux à la problématique.

La maquette devra répondre aux critères suivants :

Critère 1 : Faire fonctionner un capteur de température et afficher la température sur la calculatrice.

Critère 2 : Faire fonctionner un capteur magnétique qui détermine si le pôle sud d'un champ magnétique est proche du capteur. Il affichera si l'aimant est présent ou non.

Critère 3 : Faire fonctionner un servomoteur pendant 1 seconde dans un sens puis 1 seconde dans le sens inverse.

Critère 4 : Faire fonctionner un servomoteur en fonction du capteur de température et en fonction du capteur de champ magnétique (par exemple si la température est trop élevée et si l'aimant est présent).

.

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Proposition de résolution

Critère 1 : Faire fonctionner un capteur de température et afficher la température sur la calculatrice.

- Bibliothèques nécessaires (voir importation de bibliothèques ci-après).

```
ÉDITEUR : VENTI
LIGNE DU SCRIPT 0001
# Projets STEM Hub
from ti_system import *
from time import *
from temperat import *

def temp():
  disp_clr()
  print("appuyer sur annul pour
    arreter")
  a=temperature("IN 1")
  while not escape():
    b=a.measurement()
    disp_at(3,"temperature(C)="+
      str(b),"left")
    sleep(0.5)
```

- L'instruction **a=temperature("IN 1")** permet d'associer la variable **a** au capteur de température branché sur le **port 1** du hub.
- L'instruction **disp_clr( )** permet d'effacer l'écran de la calculatrice.
- L'instruction **b=a.mesurement( )** permet de lire la valeur du capteur de température et de stocker cette valeur dans la variable **b**.
- L'instruction **Disp_at(3,T,"left")** permet l'affichage de la variable **T** à la ligne **3** et à **gauche** sur l'écran de la calculatrice. Pour rappel, il y a 8 lignes et 16 colonnes sur l'écran.
- L'instruction **sleep (0.5)** permet de temporiser la boucle.

Critère 2 : Faire fonctionner un capteur magnétique qui détermine si le pôle sud d'un champ magnétique est proche du capteur. Il affichera si l'aimant est présent ou non.

- Bibliothèque nécessaire (voir importation de bibliothèques ci-après).

```
ÉDITEUR : VENTI
LIGNE DU SCRIPT 0024
from magnetic import *
def aimant():
  print("appuyer sur annul pour
    arreter")
  c=magnetic("IN 3")
  while not escape():
    d=c.measurement()
    print(d)
    if d<100:
      print("aimant present")
    else:_
      print("aimant non present"
      )
    sleep(0.5)
```

- Il faudra être vigilant sur le sens de l'aimant !
- L'instruction **c=magnetic("IN 3")** permet d'associer la variable **c** au capteur magnétique branché sur le **port 3** du hub.
- L'instruction **d=c.mesurement( )** permet de lire la valeur du capteur magnétique et de stocker cette valeur dans la variable **d**.

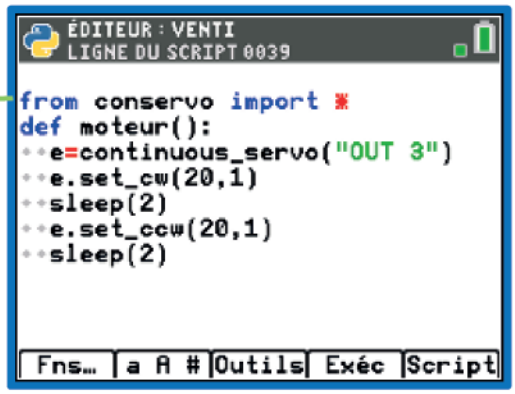
Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Fiche méthode

Critère 3 : Faire fonctionner un servomoteur .

- Bibliothèque nécessaire (voir importation de bibliothèques ci-après).

- L'instruction **e.set_cw(20,1)** permet de faire tourner le moteur dans un sens avec une vitesse égale à 20 pendant 1s. La vitesse peut varier entre 0 et 100. La commande **ccw** fait tourner le moteur dans le sens inverse.
- Attention : une alimentation externe est indispensable pour faire fonctionner le servomoteur.

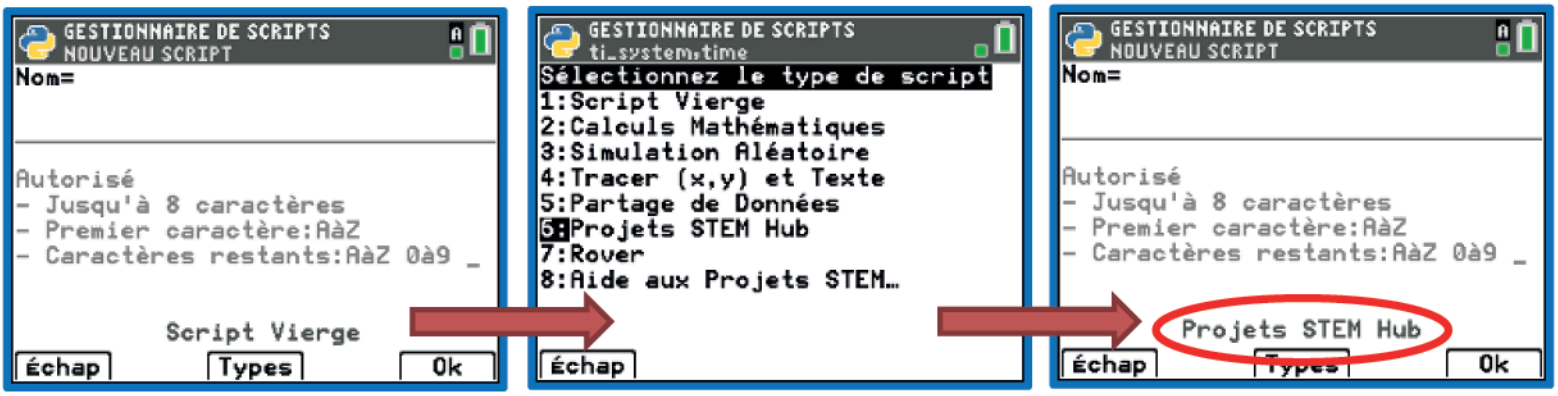


Critère 4 : Faire fonctionner un servomoteur en fonction du capteur de température et en fonction du capteur de champ magnétique (par exemple si la température est trop élevée et si l'aimant est présent).

Regrouper les scripts précédents en un seul script.

Importation des bibliothèques

- Lors de la création du script, sélectionner la rubrique **Types**, puis **Projet STEM Hub**. Cela permettra de charger automatiquement les deux bibliothèques suivantes : *time* et *ti_systeme*.



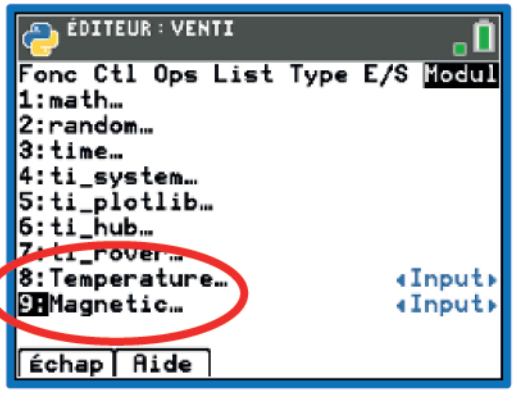
Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Fiche méthode

- Une fois le nouveau script créé, pour importer la bibliothèque *Temperature* appuyer sur la touche f(x), sélectionner la rubrique **Modul**, puis suivre les copies d'écran suivantes :

- Pour utiliser la bibliothèque *Temperature* importée, appuyer sur la touche f(x), sélectionner la rubrique **Modul**, puis sélectionner **8 : Temperature…**

- Au fur et à mesure que les bibliothèques sont importées, elles s'ajoutent aux bibliothèques déjà présentes (*math*, *random*, …etc.) et permettent d'accéder aux instructions qu'elles contiennent.

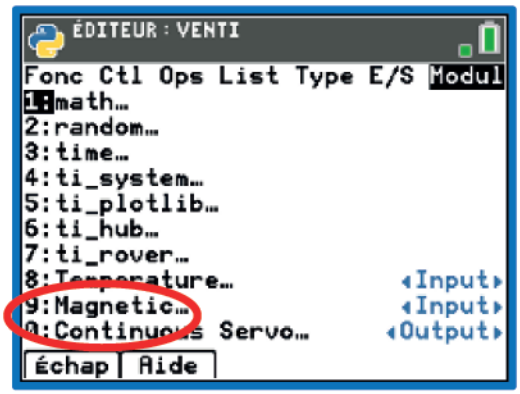


- Pour importer la bibliothèque *Magnetic* appuyer sur la touche f(x), sélectionner la rubrique **Modul**, puis suivre les copies d'écran suivantes :

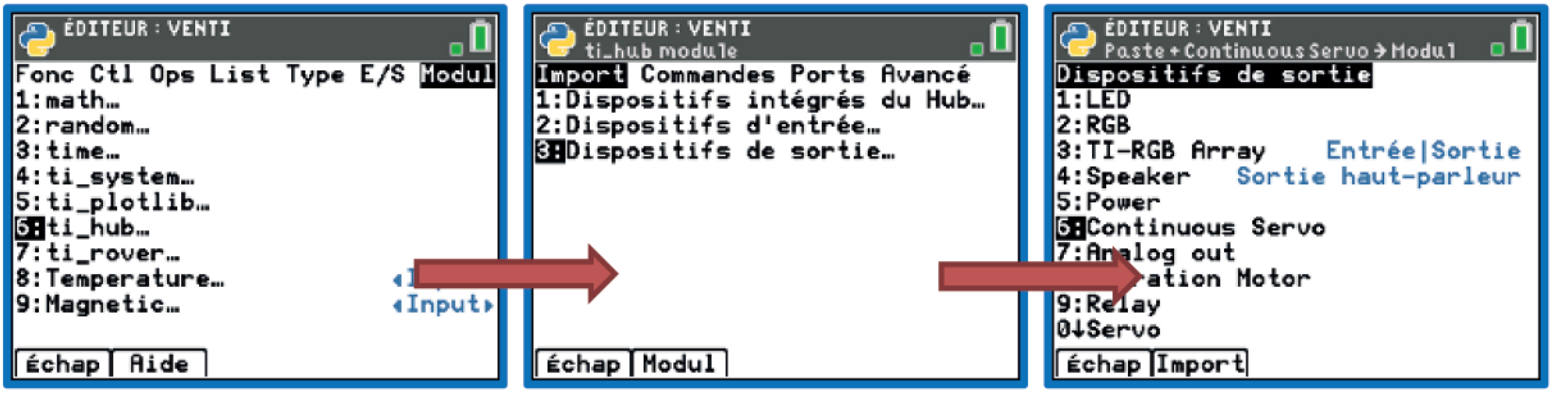
Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Fiche méthode

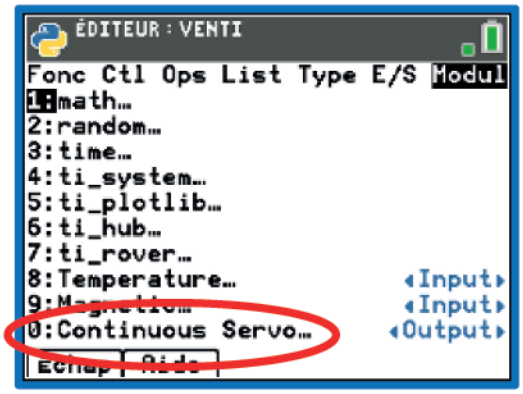
- Pour utiliser la bibliothèque *Magnetic* importée, appuyer sur la touche f(x), sélectionner la rubrique **Modul**, puis sélectionner **9 : Magnetic…**



- Pour importer la bibliothèque *Continous Servo* appuyer sur la touche f(x), sélectionner la rubrique **Modul** puis suivre les copies d'écran suivantes :



- Pour utiliser la bibliothèque *Continuous* importée, appuyer sur la touche f(x), sélectionner la rubrique **Modul** puis sélectionner **0 : Continuous Servo…**



Remarque : le nombre, l'ordre et la numérotation des modules dans ce menu dépend évidemment du nombre de modules importés et de l'ordre dans lequel ils ont été importés.

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

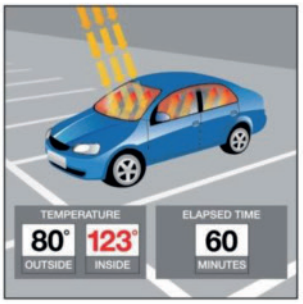
Alarme pour sauver des vies

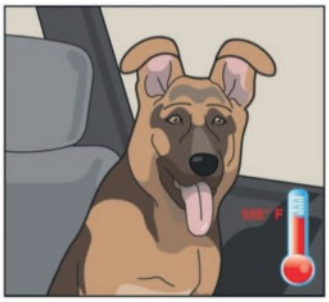
Compétences visées

Un des objectifs de l'enseignement de SNT est de développer et de coder des scripts PYTHON afin d'apporter une réponse à une problématique précise. A travers le thème " informatique embarquée et objets connectés", nous pouvons notamment travailler les compétences suivantes dans l'activité proposée :

- Coder des scripts simples d'acquisition de données.
- Gérer des entrées/sorties à travers les ports utilisés par le système.
- Écrire et développer des algorithmes pour résoudre une problématique.
- Identifier des algorithmes de contrôle des comportements physiques à travers les données des capteurs.

Situation déclenchante





Estimated Vehicle Interior Air Temperature v. Elapsed Time

Elapsed Time	Outside Air Temperature (F)					
	70	75	80	85	90	95
0 minutes	70	75	80	85	90	95
10 minutes	89	94	99	104	109	114
20 minutes	99	104	109	114	119	124
30 minutes	104	109	114	119	124	129
40 minutes	108	113	118	123	128	133
50 minutes	111	116	121	126	131	136
60 minutes	113	118	123	128	133	138
> 1 hour	115	120	125	130	135	140

Inside Air Temperature (F)

Chaque été, des drames se produisent concernant les enfants et les animaux de compagnie laissés dans des voitures exposées au soleil. Les conséquences peuvent aller du simple "coup de chaleur" jusqu'au décès, dans les pires des cas. L'intérieur d'une voiture se réchauffe beaucoup plus vite que l'extérieur en raison de l'effet de serre. Les rayons du soleil pénètrent dans le véhicule à travers les vitres et frappent la surface de la voiture. Cette lumière visible est absorbée et redirigée sous forme de lumière infrarouge. Le rayonnement infrarouge possède une longueur d'onde plus grande que la lumière visible ; il ne peut pas s'échapper par les fenêtres. Le rayonnement piégé fait ainsi monter la température à l'intérieur de la voiture plus rapidement que la température extérieure.

Problématique

Comment concevoir une solution technologique pour protéger les occupants des voitures d'une chaleur excessive ?

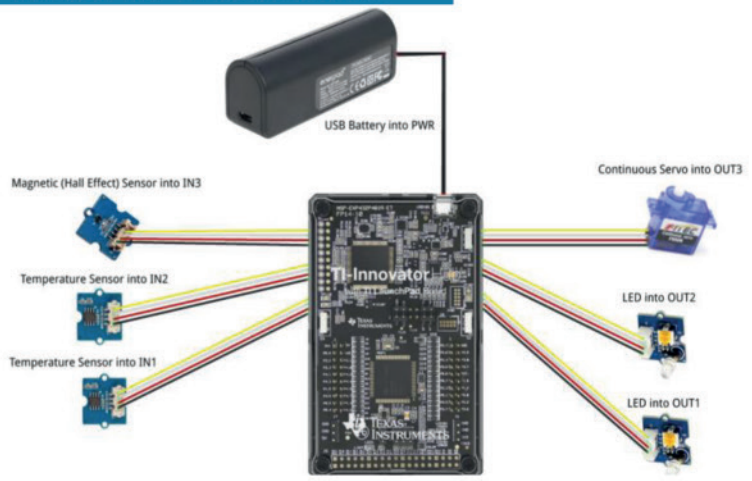
Thème : informatique embarquée et objets connectés

Classe enseignée : SNT

Fiche méthode

L.DIDIER

Matériel nécessaire



- Calculatrice TI-83 Premium CE
- Câble (calculatrice/Hub)
- TI-Innovator Hub
- Câble x 6
- Grove Temperature sensor x 2
- Hall effect magnetic proximity sensor
- LED blanche x2
- Moteur
- USB Battery + Cable
- Modèle réduit, boîte à chaussures ou autre pour figurer une voiture

Déroulement possible du projet

Travail de groupe possible. Chaque groupe disposera du matériel ci-dessus et devra concevoir une solution à la problématique sous forme d'une maquette munie d'une documentation qui explique les scripts pilotant la maquette. A la fin du projet, les groupes pourront voter pour élire la production qui répond le mieux à la problématique.
La maquette devra répondre aux critères suivants :

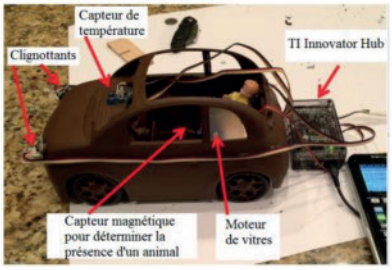
Critère 1: Jouer 2 sons de 1 seconde chacun dans une boucle pour simuler une alarme sonore.

Critère 2 : Faire clignoter 2 leds externes pour simuler l'allumage des clignotants.

Critère 3 : Faire fonctionner un servomoteur pour l'ouverture des vitres.

Critère 4 : Faire fonctionner deux capteurs de température (température intérieure et extérieure) et afficher les températures sur la calculatrice.

Critère 5 : Faire fonctionner un capteur magnétique qui détermine si le pôle sud d'un champ magnétique est proche du capteur. Il affichera si l'aimant est présent ou non.
Critère final : Regrouper les compétences acquises lors des étapes précédentes pour développer un système d'alarme pour un modèle de voiture déterminant si un animal domestique est présent (aimant) et si la température à l'intérieur de la voiture atteint un seuil critique avant de déclencher l'alarme et d'ouvrir les fenêtres.



Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Thème : informatique embarquée et objets connectés

Classe enseignée : SNT

Fiche méthode

L.DIDIER

Proposition de résolution

Pour obtenir des détails sur les critères de 1 à 5 consulter les fiches "ventilateur autonome" et "alarme de voiture".

- Importation des bibliothèques (voir le paragraphe importation des bibliothèques ci-après)

- Association des différentes variables aux différents capteurs.

- L'instruction **disp_at(3,T,"left")** permet l'affichage de la chaîne de caractères T à la ligne 3 et à gauche sur l'écran de la calculatrice. Pour rappel, il y a 8 lignes et 16 colonnes sur l'écran.

- Création de la variable w. Elle sera égale à 1 si la fenêtre est fermée et 0 sinon. Il est également possible d'utiliser un booléen (True ou False).

- Lecture des capteurs et stockage des différentes valeurs.

- Affichage des différentes informations .

- Déclenchement de l'alarme sonore,allumage des diodes et ouverture de la fenêtre.

```
ÉDITEUR : ALARMEVO
LIGNE DU SCRIPT 0001
_ Projets STEM Hub
from ti_system import *
from time import *
from temperat import *
from magnetic import *
from conservo import *
from led import *
import sound

def alarmevoiture():
  a=temperature("IN 1")
  b=temperature("IN 2")
  c=led("OUT 1")
  d=led("OUT 2")
  e=continuous_servo("OUT 3")
  f=magnetic("IN 3")

  disp_clr()
  disp_at(1,"alarme actiee","left")
  disp_at(9,"appuyer sur annul_pour arreter","left")

  w=1
  while not escape():
    temp1=a.measurement()
    temp2=b.measurement()
    mag=f.measurement()
    if mag<100:
      disp_at(4,"presence detectee","left")
    else:
      disp_at(4,"aucune presence","left")
    if w==1:
      disp_at(5,"fenetre fermee","left")
    else:
      disp_at(5,"fenetre ouverte","left")
    disp_at(6,"temp int="+str(temp1),"left")
    disp_at(7,"temp ext="+str(temp2),"left")
    if temp1>25 and mag<100:
      disp_at(2,"pesonne en danger","left")
      c.blink(3,1)
      d.blink(3,1)
      sound.tone(440,0.5)
      sleep(0.5)
      sound.tone(880,0.5)
      sleep(0.5)
      if w==1:
        e.set_cw(30,0.25)
```

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Thème : informatique embarquée et objets connectés

Classe enseignée : SNT

Fiche méthode

L.DIDIER

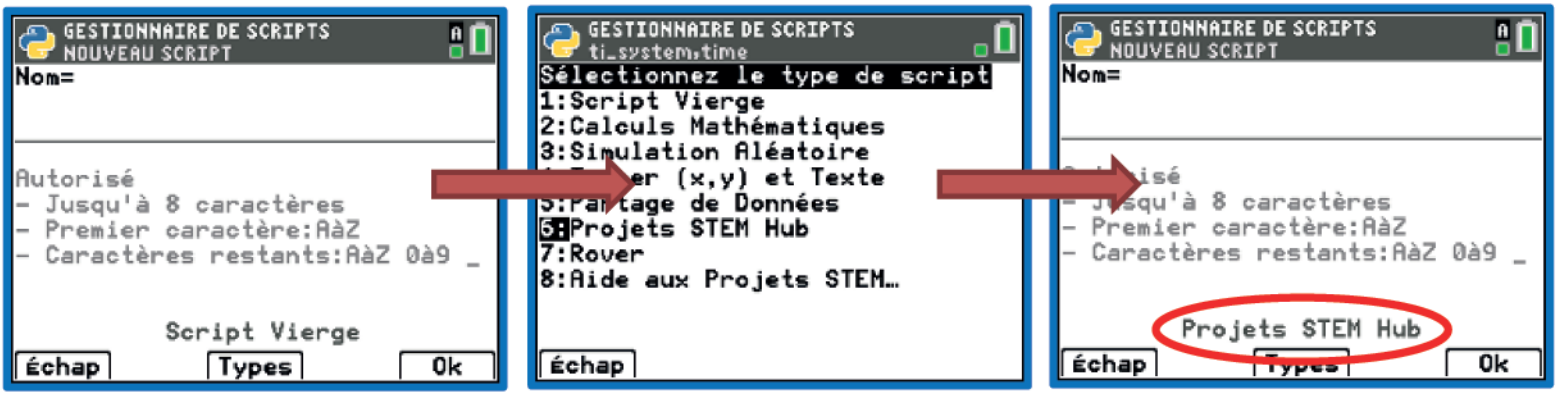
- Fermeture de la fenêtre
- Arrêt de l'alarme

```
        sleep(0.25)
        w=0
    else:
        c.off()
        d.off()
        disp_at(2,"aucun probleme
    detecte","left")
        if w==0:
            e.set_ccw(30,0.25)
            sleep(0.25)
            w=1
c.off()
d.off()
disp_clr()
disp_at(1,"arret alarme","left
    ")
```

Fns… | a A # | Outils | Exéc | Script

Importation des bibliothèques

- Lors de la création du script, sélectionner la rubrique Types, puis Projet STEM Hub. Cela permettra de charger automatiquement les deux bibliothèques suivantes : *time* et *ti_systeme*.



- Pour l'importation des autres bibliothèques, reportez-vous aux fiches "ventilateur autonome" et "alarme de voiture".

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Tracé de polygones réguliers

Compétences visées

Un des objectifs de l'enseignement de SNT est de développer et de coder des scripts PYTHON afin d'apporter une réponse à une problématique précise. A travers le thème " informatique embarquée et objets connectés", nous pouvons notamment travailler les compétences suivantes dans l'activité proposée :

- Écrire et développer des algorithmes pour répondre à des problèmes.
- Faire preuve d'autonomie, d'initiative et de créativité.
- Coopérer au sein d'une équipe.
- Rechercher de l'information, apprendre à utiliser des sources de qualité, partager des ressources.
- Présenter un problème ou sa solution, développer une argumentation dans le cadre d'un débat.

Situation déclenchante

Dans la vie de tous les jours, nous sommes entourés de formes géométriques : bâtiments, œuvres d'art, …etc. Dans certains cas, il s'agit de polygones réguliers. Mais qu'est-ce qu'un polygone régulier ? Comment le tracer ?

Problématique

Comment programmer le robot Rover pour qu'il puisse tracer un polygone régulier au sol et sur l'écran de la calculatrice, connaissant le nombre de côtés et la longueur d'un côté ?

Fiche méthode

Matériel nécessaire

- Calculatrice TI-83 Premium CE
- Câble (calculatrice/Hub)
- TI-Innovator Hub
- Rover
- Marqueur

Déroulement possible du projet

Travail de groupe possible. Chaque groupe disposera du matériel ci-dessus et devra concevoir une réponse à la problématique sous forme d'une démonstration munie d'une documentation qui explique les scripts PYTHON pilotant le rover. A la fin du projet, les groupes pourront voter pour élire la production qui répond le mieux à la problématique.

La production devra répondre aux critères suivants :

Critère 1 : Définir succinctement un polygone régulier et ses principales propriétés.

Critère 2 : Le rover devra tracer un polygone au sol connaissant le nombre de côtés et la longueur d'un côté.

Critère 3 : Tracer les déplacements du Rover sur l'écran de la calculatrice.

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Fiche méthode

Proposition de résolution

La fonction polygone prend comme arguments la longueur d'un côté (notée l) et le nombre de côtés du polygone régulier (noté c).

Importation des bibliothèques nécessaires (voir remarques).

Instructions de tracé sur la calculatrice :
- L'instruction **plt.cls()** permet d'effacer l'écran.
- L'instruction **plt.axes("on")** permet d'afficher les axes.
- L'instruction **plt.title("polygone regulier a"+str(c)+"cotés")** permet d'afficher le titre.
- L'instruction **plt.pen("medium,"solid")** permet de modifier le style de tracé.
- L'instruction **plt.grid(1,1,"dot")** permet d'afficher une grille.

Les instructions de déplacement.

L'instruction **rv.wait_until_done( )** permet d'attendre la fin du déplacement précédent de manière à synchroniser le tracé sur la calculatrice avec le tracé au sol.

L'instruction **rv.waypoint_x( )** permet de récupérer l'abscisse de la fin du déplacement.

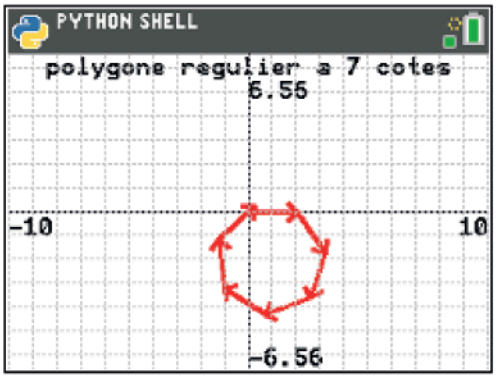
L'instruction **plt.line(x,y,X,Y,"arrow")** permet de tracer une flèche du point de coordonnées (x,y) au point de coordonnées (X,Y).

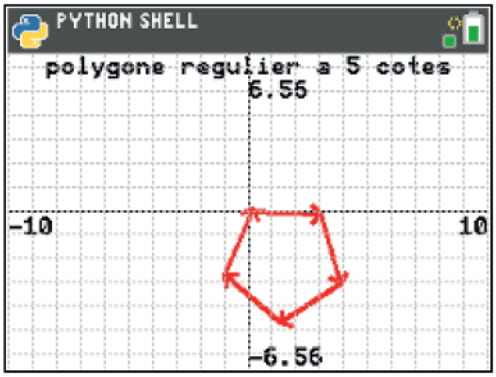
L'instruction **plt.show_plot( )** permet d'afficher le graphique.

```
ÉDITEUR : POLYGONE
LIGNE DU SCRIPT 0001
# TI-Rover
from time import *
from ti_system import *
import ti_rover as rv
import ti_plotlib as plt

def polygone(l,c):
  plt.cls()
  plt.axes("on")
  plt.pen("medium","solid")
  plt.color(255,0,0)
  plt.grid(1,1,"dot")
  plt.title("polygone regulier a
      "+str(c)+" cotes")
  x=0
  y=0
  b=360/c
  for i in range(c):
    rv.forward(l)
    rv.right(b)
    rv.wait_until_done()
    X=rv.waypoint_x()
    X=rv.waypoint_x()
    Y=rv.waypoint_y()
    plt.line(x,y,X,Y,"arrow")
    x=X
    y=Y
  plt.show_plot()
  rv.disconnect_rv()
_
Fns… | a A # | Outils | Exéc | Script
```

Exemples de représentation graphique obtenues :



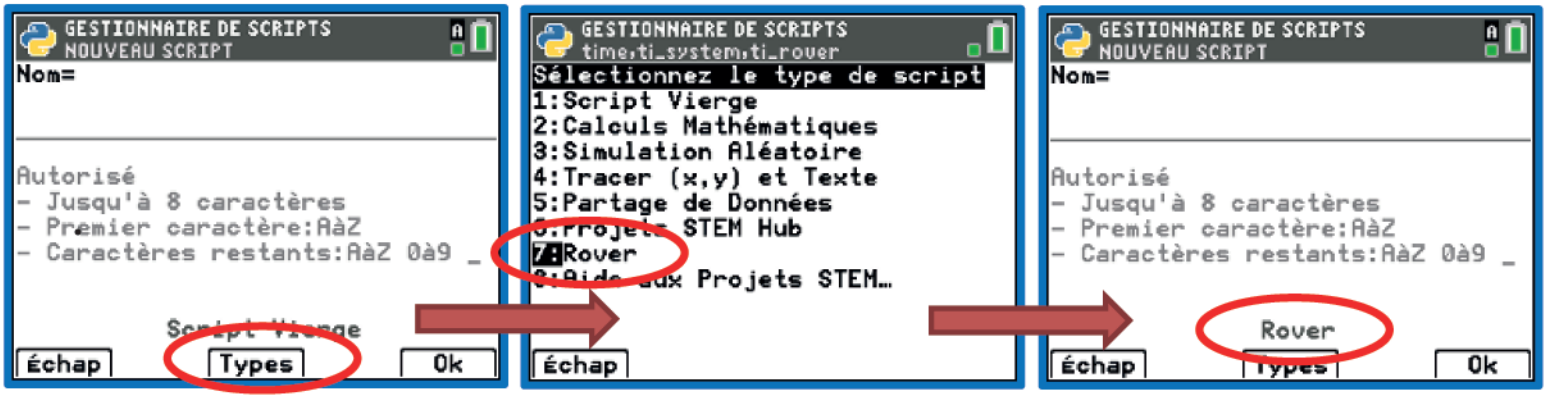


Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

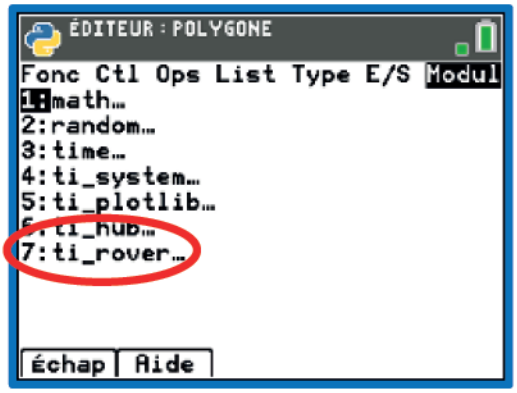
Fiche méthode

Utilisation des bibliothèques

- Lors de la création du script, sélectionner la rubrique Types, puis Rover. Cela permettra de charger automatiquement les trois bibliothèques suivantes : *time* , *ti_systeme* et *ti_rover*.

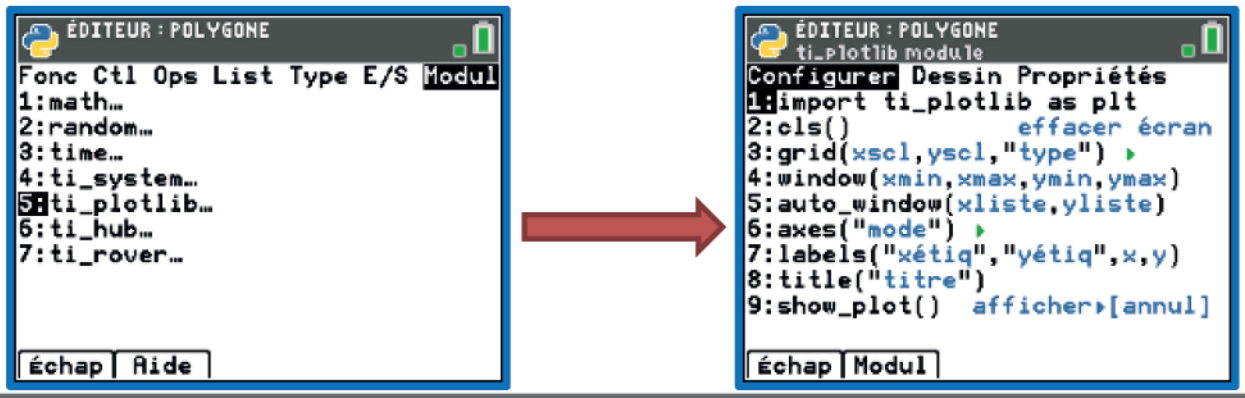


- Pour utiliser la bibliothèque *ti_rover* importée, appuyer sur la touche f(x), sélectionner la rubrique Modul, puis sélectionner **7 : ti_rover…**



- Au fur et à mesure que les bibliothèques sont importées, elles s'ajoutent aux bibliothèques déjà présentes (*math*, *random*, …etc.) et permettent d'accéder aux instructions qu'elles contiennent.

- Une fois le nouveau script créé, pour importer la bibliothèque *ti_plotlib*, appuyer sur la touche f(x), sélectionner la rubrique Modul, puis suivre les copies d'écran suivantes :



Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Thème : informatique embarquée et objets connectés

Classe enseignée : SNT

L.DIDIER

Alarme de voiture

Compétences visées

Un des objectifs de l'enseignement de SNT est de développer et de coder des scripts PYTHON afin d'apporter une réponse à une problématique précise. A travers le thème " informatique embarquée et objets connectés", nous pouvons notamment travailler les compétences suivantes dans l'activité proposée :

- Coder des scripts simples d'acquisition de données.
- Gérer des entrées/sorties à travers les ports utilisés par le système.
- Écrire et développer des algorithmes pour résoudre une problématique.
- Identifier des algorithmes de contrôle des comportements physiques à travers les données des capteurs.

Situation déclenchante

Aujourd'hui, de nombreuses voitures sont équipées d'alarmes. Ces alarmes ne sont pas nécessairement associées à la défense du véhicule. Qu'il s'agisse d'une alarme sonore ou d'une alarme visuelle, une alarme peut être un simple indicateur d'un événement particulier.

Problématique

Comment concevoir une solution technologique pour simuler le déclenchement d'une alarme de voiture ?

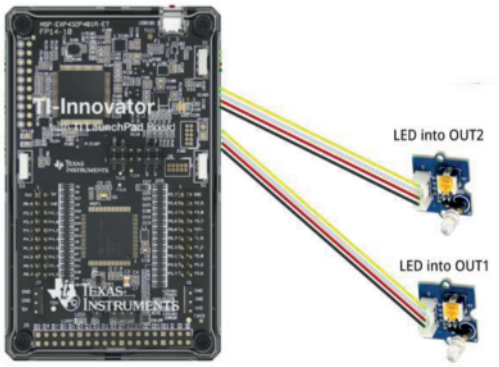
Thème : informatique embarquée et objets connectés

Classe enseignée : SNT

Fiche méthode

L.DIDIER

Matériel nécessaire



- Calculatrice TI-83 Premium CE
- Câble (calculatrice/Hub)
- TI-Innovator Hub
- Câble x2
- LED blanche x2

Déroulement possible du projet

Travail de groupe possible. Chaque groupe disposera du matériel ci-dessus et devra concevoir une réponse à la problématique sous forme d'une maquette munie d'une documentation qui explique les scripts pilotant la maquette. A la fin du projet, les groupes pourront voter pour élire la production qui répond le mieux à la problématique.

La maquette devra répondre aux critères suivants :

Critère 1 : Jouer 2 sons de 1 seconde chacun dans une boucle pour simuler une alarme sonore.

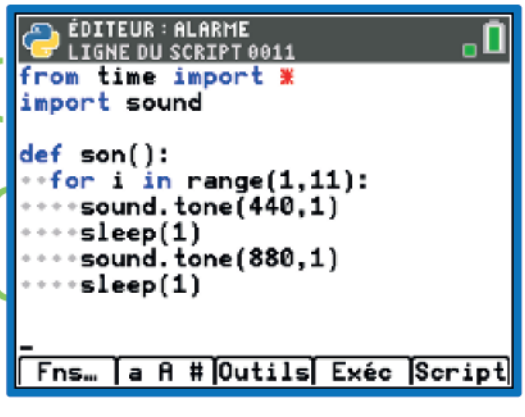
Critère 2 : Faire clignoter 2 leds externes pour simuler l'allumage des clignotants.

Proposition de résolution

Critère 1 : Jouer 2 sons de 1 seconde chacun dans une boucle pour simuler une alarme sonore.

- Importation des bibliothèques *sound* et *time* (voir le paragraphe importation des bibliothèques ci-après).

- L'instruction **sound.tone (440,1)** permet d'émettre un son à une fréquence de 440 hertz pendant 1 seconde.
- L'instruction **sleep (1)** permet de temporiser (1s) l'exécution du programme pendant l'émission du son précédent.



Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Thème : informatique embarquée et objets connectés

Classe enseignée : SNT

Fiche méthode

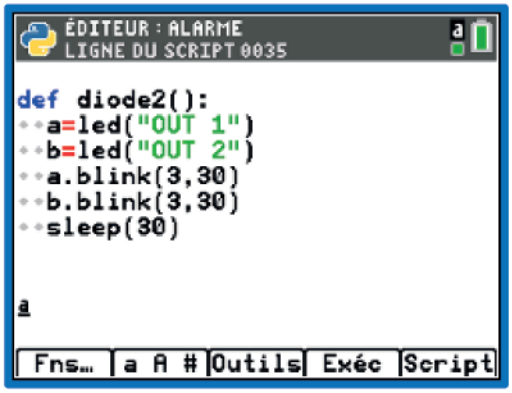
L.DIDIER

Critère 2 : Faire clignoter 2 leds externes.

- Importation de la bibliothèque *led* (voir remarque).
- L'instruction : **a=led("OUT 1")** permet d'associer la variable a, à la led (objet physique) connectée au port OUT 1 du hub.
- L'instruction : **b=led("OUT 2")** permet d'associer la variable b, à la led (objet physique) connectée au port OUT 2 du hub.
- L'instruction **a.on( )** , permet d'allumer la led branchée au port 1.
- L'instruction **a.off( )** , permet d'éteindre la led branchée au port 1.

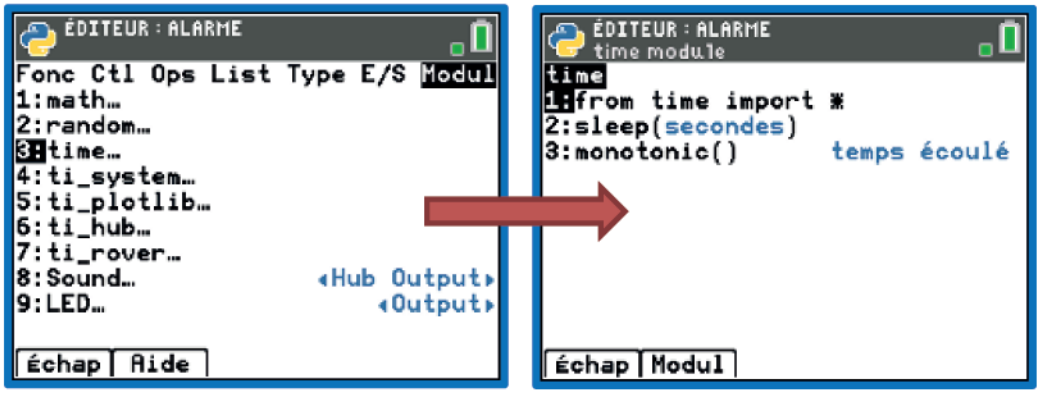
Critère 2 (seconde version) : Faire clignoter 2 leds externes.

- L'instruction : **a.blink(3,30)** permet d'allumer la led branchée au port 1 et de la faire clignoter 3 fois par seconde pendant 30 secondes.
- L'instruction **sleep(30)** permet de temporiser le programme le temps de l'exécution de l'instruction précédente (le clignotement des leds).



Importation des bibliothèques

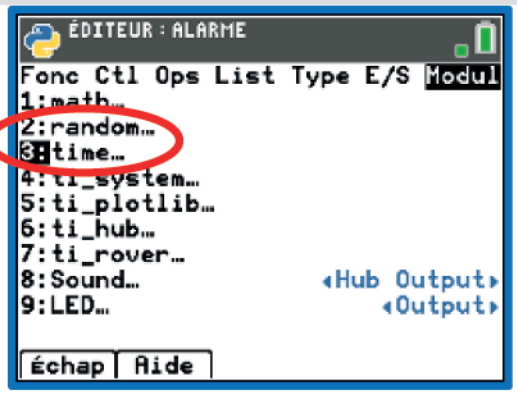
- Une fois le nouveau script créé, pour importer la bibliothèque *time* appuyer sur la touche f(x), sélectionner la rubrique Modul, puis suivre les copies d'écran suivantes :



Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus !

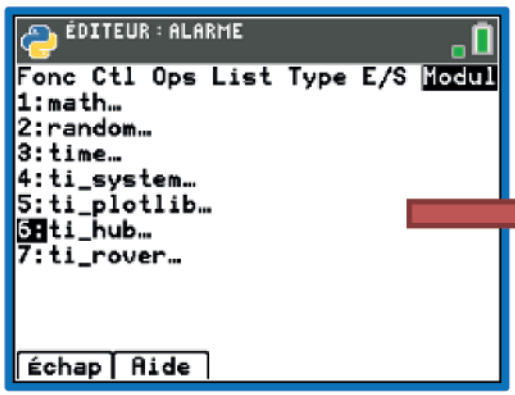
Fiche méthode

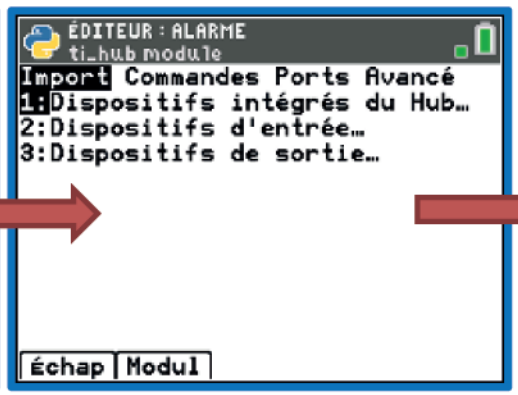
- Pour utiliser la bibliothèque *time* importée, appuyer sur la touche f(x), sélectionner la rubrique Modul puis sélectionner **3 : time**…



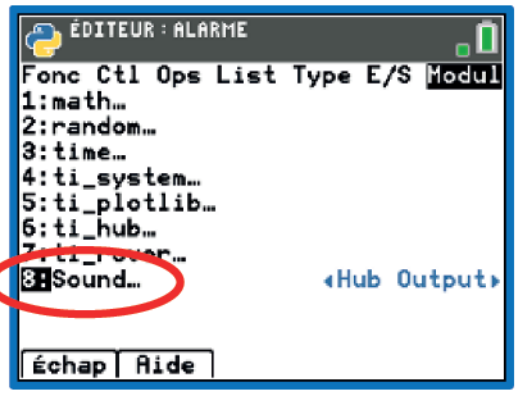
- Au fur et à mesure que les bibliothèques sont importées, elles s'ajoutent aux bibliothèques déjà présentes (math, random, etc.) et permettent d'accéder aux instructions qu'elles contiennent.

- Une fois le nouveau script créé, pour importer la bibliothèque *sound,* appuyer sur la touche f(x), sélectionner la rubrique Modul, puis suivre les copies d'écran suivantes :





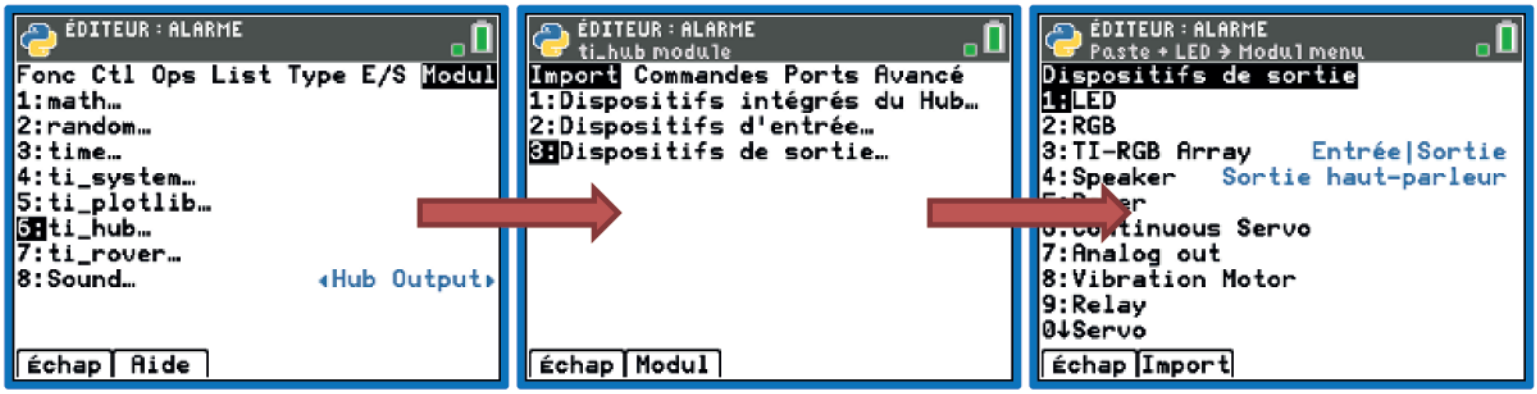
- Pour utiliser la bibliothèque *sound* importée , appuyer sur la touche f(x), sélectionner la rubrique Modul, puis sélectionner **8 : Sound**…



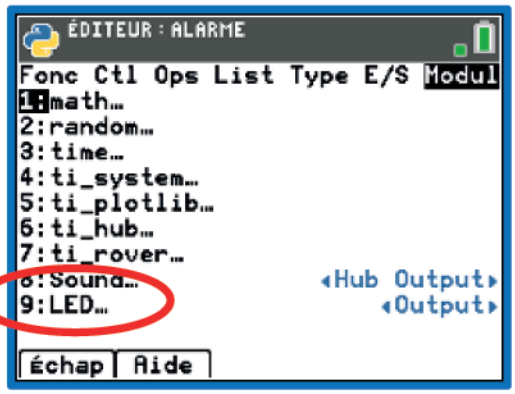
Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Fiche méthode

- Pour importer la bibliothèque *led*, appuyer sur la touche f(x), sélectionner la rubrique Modul, puis suivre les copies d'écran suivantes :



- Pour utiliser la bibliothèque *led* importée, appuyer sur la touche f(x), sélectionner la rubrique Modul, puis sélectionner **9 : LED…**



Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus !

Thème : informatique embarquée et objets connectés

Classe enseignée : SNT

L.DIDIER

Park Assist

Compétences visées

Un des objectifs de l'enseignement de SNT est de développer et de coder des scripts PYTHON afin d'apporter une réponse à une problématique précise. A travers le thème " informatique embarqué et objets connectés" , nous pouvons notamment travailler les compétences suivantes dans l'activité proposée :

- Coder des scripts simples d'acquisition de données.
- Gérer des entrées/sorties à travers les ports utilisés par le système.
- Écrire et développer des algorithmes pour résoudre une problématique.
- Identifier des algorithmes de contrôle des comportements physiques à travers les données des capteurs.

Le développement des logiciels embarqués est délicat, car il pose souvent des questions de temps-réel, c'est-à-dire de respect de temps de réponse imposé. Ceci conduit à des méthodes de programmation spécifiques dans certains cas.

Situation déclenchante

Aujourd'hui, de nombreuses voitures sont équipées de système de Park Assist. Comment fonctionne ce système et comment la technologie permet de nous assister lors de situations compliquées ?

Problématique

Comment concevoir une solution technologique
pour détecter une place et simuler un Park Assist ?

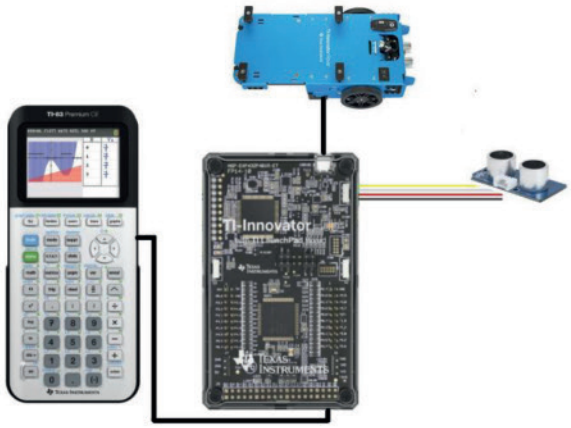
Thème : informatique embarquée et objets connectés

Classe enseignée : SNT

Fiche méthode

L.DIDIER

Matériel nécessaire

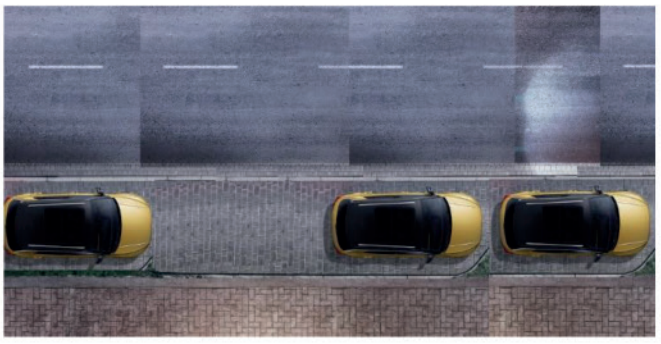


- Calculatrice TI-83 Premium CE
- Câble (calculatrice/Hub)
- TI-Innovator Hub
- Rover
- Capteur de distance (ranger)
- Câbles

Déroulement possible du projet

Travail de groupe possible. Chaque groupe disposera du matériel ci-dessus et devra concevoir une réponse à la problématique sous forme d'une démonstration munie d'une documentation qui explique les scripts PYTHON pilotant le Rover. A la fin du projet, les groupes pourront voter pour élire la production qui répond le mieux à la problématique.

Les scripts développés devront répondre à la problématique uniquement dans la situation ci-contre et répondre aux critères suivants :



Critère 1 : Le rover devra circuler sur la route, à moins de 20 cm latéralement des voitures garées.

Critère 2 : Détecter une place suffisante pour garer le rover.

Critère 3 : Mener une manœuvre pour garer le rover .

Critère 4 : Allumer la diode du rover lorsque la manœuvre est engagée.

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Thème : informatique embarquée et objets connectés

Classe enseignée : SNT

Fiche méthode

L.DIDIER

Proposition de résolution

Il faudra fixer le ranger sur le côté du rover de manière à pouvoir détecter une place latérale lors du déplacement du rover (voir vidéo).

Importation des bibliothèques (voir le paragraphe importation des bibliothèques)

L'instruction **rv.forward(50)** permet d'avancer de 50 unités de mesure (le décimètre, par défaut).

Tant que le rover ne détecte pas de place latérale il continue à avancer.

L'instruction **a=ranger("IN1")** permet d'associer le ranger à la variable a.

L'instruction **b=a.measurement( )** permet de lire la distance latérale et permet de la stocker dans b. Cette distance est en mètres.

Détection d'une place et calcul de la longueur de la place.

L'instruction **d=rv.waypoint_distance( )** permet de

connaitre la distance parcourue depuis l'instruction **rv.path_clear( )**.

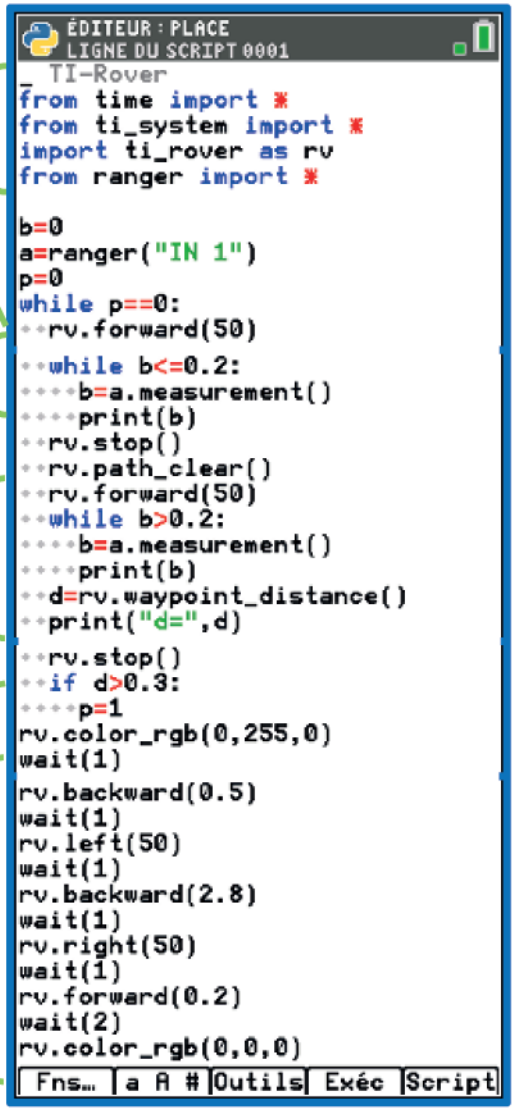
Vérification si la distance est suffisante pour garer le rover (30 cm).

Manœuvre pour garer le rover et allumer la diode RGB du rover.

L'instruction **rv.color_rgb(0,255,0)** permet d'allumer la diode RGB du rover avec la couleur verte. L'intensité est ici de 255 (et peut varier entre 0 et 255).

L'instruction **rv.left(50)** permet au rover de tourner à gauche de 50 degrés.

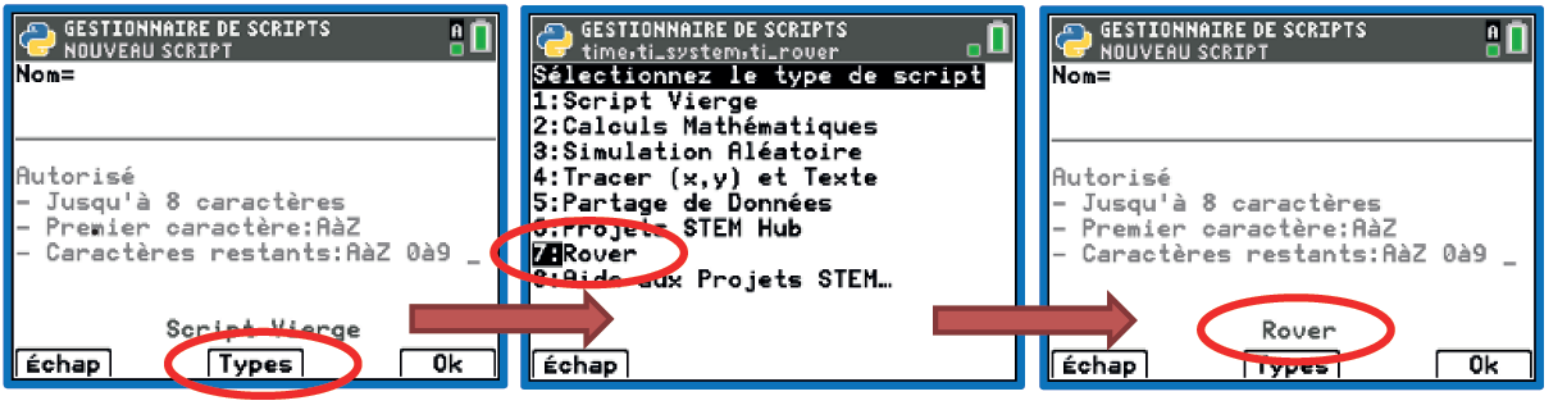
L'instruction **rv.backward(2.8)** permet au rover de reculer de 2.8 unités de mesure (le décimètre).



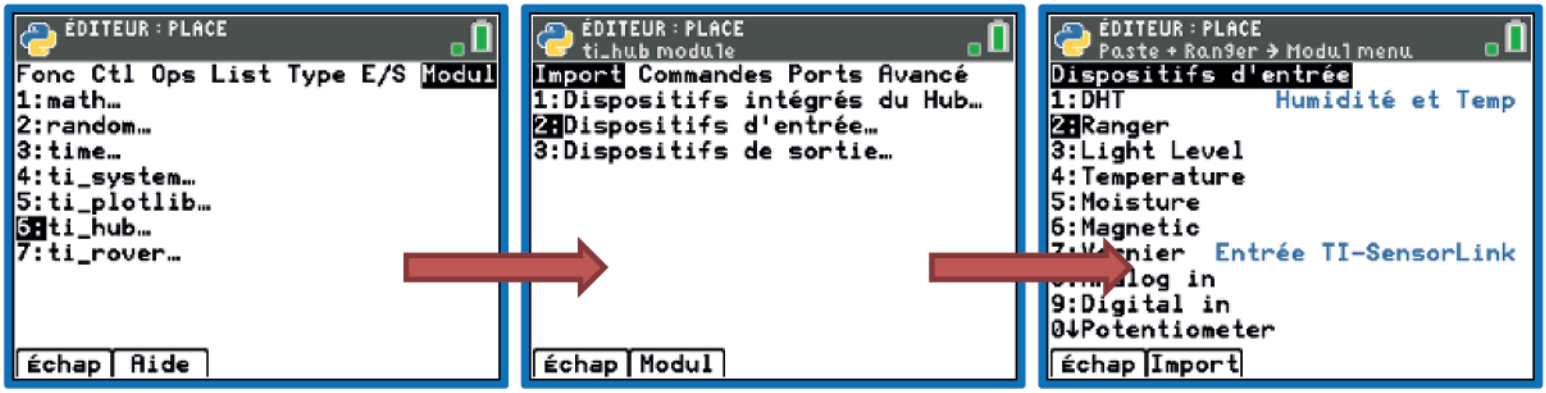
Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus !

Importation des bibliothèques

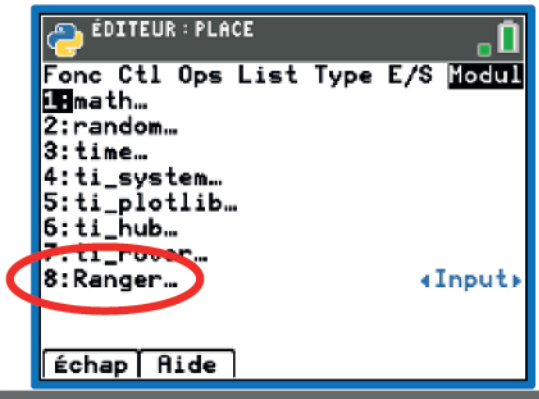
- Lors de la création du script, sélectionner la rubrique Types, puis Rover. Cela permettra de charger automatiquement les trois bibliothèques suivantes : *time* , *ti_systeme* et *ti_rover*.



- Une fois le nouveau script créé, pour importer la bibliothèque *Ranger* appuyer sur la touche f(x), sélectionner la rubrique Modul, puis suivre les copies d'écran suivantes :



- Pour utiliser la bibliothèque *Ranger* importée, appuyer sur la touche f(x), sélectionner la rubrique Modul, puis sélectionner **8 : Ranger…**
- Au fur et à mesure que les bibliothèques sont importées, elles s'ajoutent aux bibliothèques déjà présentes (*math*, *random*, …etc.) et permettent d'accéder aux instructions qu'elles contiennent.



Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus !

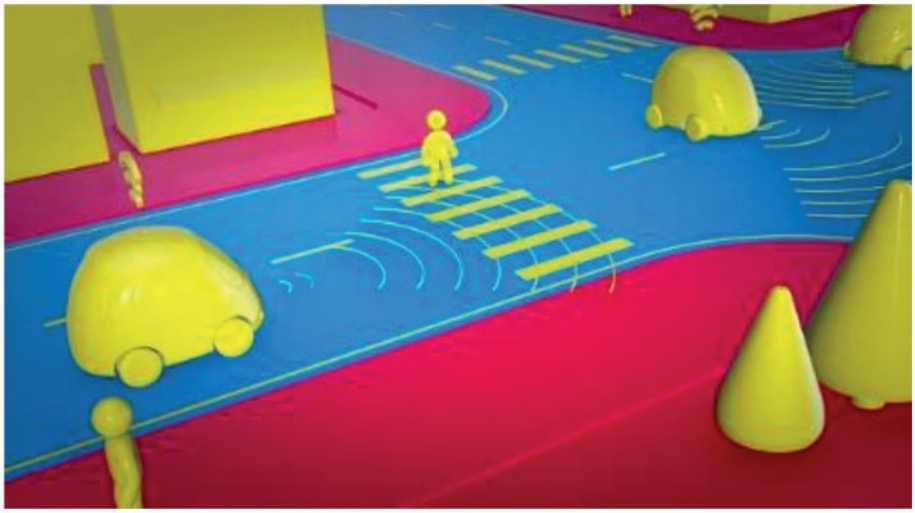
Voiture autonome

Compétences visées

Un des objectifs de l'enseignement de SNT est de développer et de coder des scripts PYTHON afin d'apporter une réponse à une problématique précise. A travers le thème " **informatique embarquée et objets connectés**", nous pouvons notamment travailler les compétences suivantes dans l'activité proposée :

- Coder des scripts simples d'acquisition de données.
- Gérer des entrées/sorties à travers les ports utilisés par le système.
- Écrire et développer des algorithmes pour résoudre une problématique.
- Identifier des algorithmes de contrôle des comportements physiques à travers les données des capteurs.

Situation déclenchante



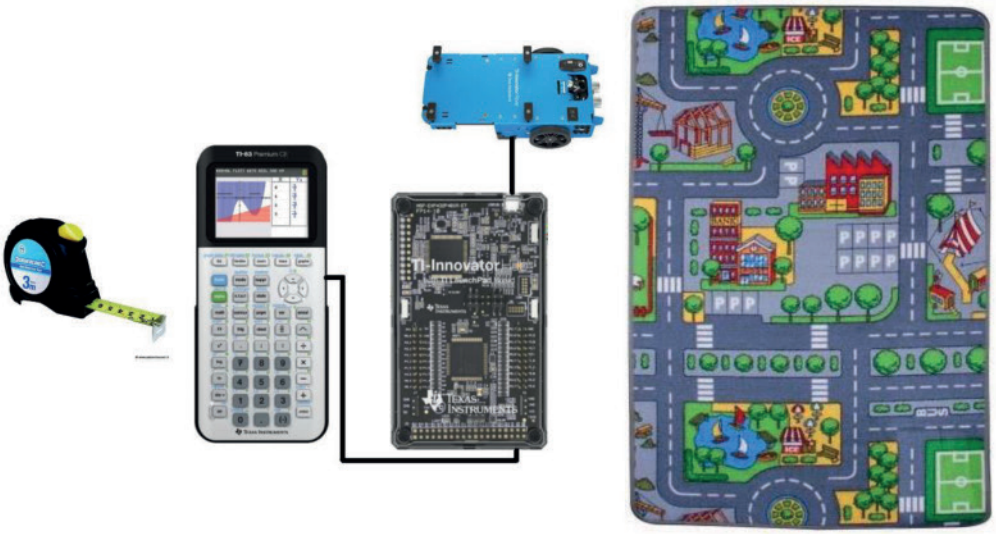
Le développement des logiciels embarqués est délicat, car il pose souvent des questions de temps-réel, c'est-à-dire de respect de temps de réponse imposé. Ceci conduit à des méthodes de programmation spécifiques.
L'impact de l'informatisation des objets devient considérable, surtout depuis que leurs interfaces s'unifient. Le but est de fabriquer des machines d'utilisation facile permettant des fonctionnalités améliorées, voire complètement nouvelles comme la voiture autonome. Celle-ci utilise à la fois des techniques de systèmes embarqués pour son fonctionnement et sa navigation et de l'intelligence artificielle pour l'analyse en temps-réel de l'environnement à l'aide de capteurs variés (caméras, radars, …etc.).

Problématique

Comment concevoir une solution technologique pour programmer le déplacement prédéfini d'un véhicule avec arrêt temporaire si un obstacle du type piéton est rencontré ?

Fiche méthode

Matériel nécessaire



- Calculatrice TI-83 Premium CE
- Câble (calculatrice/Hub)
- TI-Innovator Hub
- Rover
- Tapis de sol
- Mètre

Déroulement possible du projet

Travail de groupe possible. Chaque groupe disposera du matériel ci-dessus et devra concevoir une réponse à la problématique sous forme d'une démonstration munie d'une documentation qui explique les scripts pilotant la rover. A la fin du projet, les groupes pourront voter pour élire la production qui répond le mieux à la problématique.

La production devra répondre aux critères suivants :

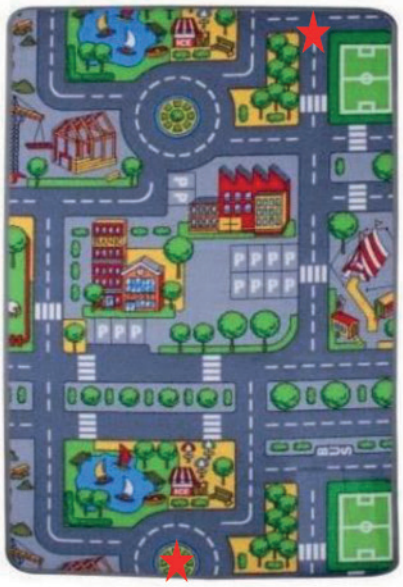
Critère 1 : Déplacer le rover en suivant les routes au sol pour aller du point de départ au point d'arrivée (exemple ci-contre).

Critère 2 : S'arrêter si un obstacle du type piéton est rencontré sur la route.

Critère 3 : Allumer la diode RGB en rouge lors du freinage si un obstacle est rencontré.

Critère 4 : Après le passage des piétons, le rover doit finir le trajet jusqu'au point d'arrivée.

Point de départ



Point d'arrivée

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Fiche méthode

Proposition de résolution

Les élèves devront dans un premier temps mesurer les déplacements à effectuer à l'aide du mètre puis faire les conversions nécessaires. L'unité de référence pour les déplacements du rover est le décimètre.

Bibliothèques nécessaires (voir remarques ci-après).

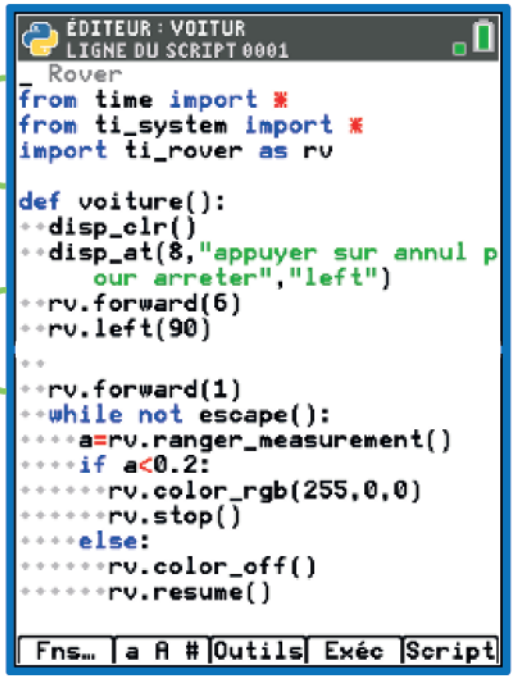
Il faut saisir l'ensemble des déplacements du rover. L'unité d'angle est le degré et l'unité de mesure est le décimètre. L'ensemble des déplacements est envoyé dans une liste. Les déplacements sont ensuite traités dans l'ordre par le rover.

L'instruction **a=rv.ranger_measurement( )** permet de lire la distance devant le rover et de la stocker dans **a**. Cette distance est en mètres.

L'instruction **rv.color_rgb(255,0,0)** permet d'allumer la diode RGB du rover avec la couleur rouge. L'intensité est ici de 255 (et peut varier entre 0 et 255).

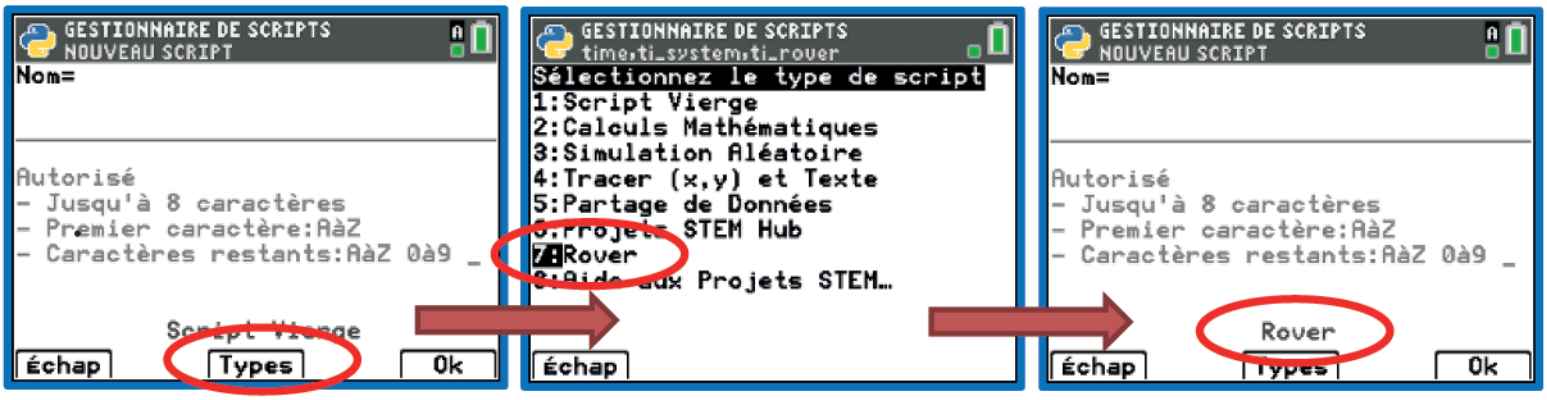
La commande **rv.stop( )** est exécutée immédiatement et n'est pas intégrée à la liste ordonnée des déplacements.

L'instruction **rv.resume( )** permet de reprendre la liste des déplacements qui avait été interrompue par la commande STOP.



Utilisation des bibliothèques

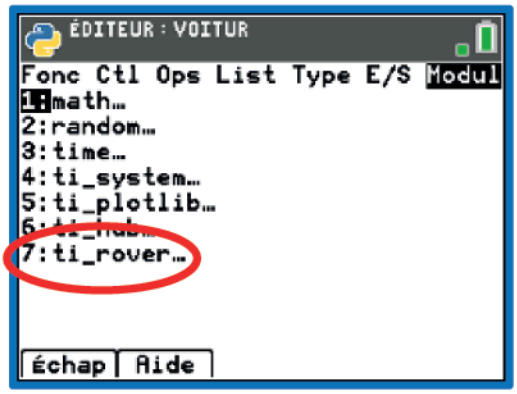
- Lors de la création du script, sélectionner la rubrique **Types**, puis **Rover**. Cela permettra de charger automatiquement les trois bibliothèques suivantes : *time*, *ti_systeme* et *ti_rover*.



Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Fiche méthode

- Pour utiliser la bibliothèque *ti_rover* importée, appuyer sur la touche f(x), sélectionner la rubrique **Modul** puis sélectionner **7 : ti_rover…**
- Au fur et à mesure que les bibliothèques sont importées, elles s'ajoutent aux bibliothèques déjà présentes (*math*, *random*, …etc.) et permettent d'accéder aux instructions qu'elles contiennent.



Compétences visées

Un des objectifs de l'enseignement de SNT est de comprendre que les requêtes menées par un moteur de recherche peuvent donner des résultats différents d'un moteur de recherche à l'autre, puisque ceux-ci utilisent des algorithmes qui leur sont propres. Cette activité vise à comprendre sur des exemples simples la manière dont on peut mesurer la popularité d'une page en utilisant le principe du "surfeur aléatoire".

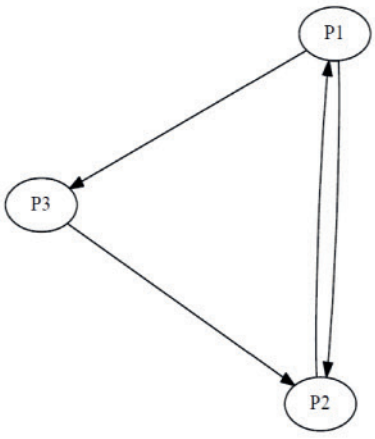
Elle s'inscrit dans la thématique "**le Web**" pour l'activité proposée : "Calculer la popularité d'une page à l'aide d'un graphe simple puis programmer l'algorithme". Nous pouvons notamment travailler les compétences suivantes dans l'activité proposée :

- Modéliser une situation par un graphe.
- Écrire et développer des algorithmes pour résoudre une problématique.

Situation déclenchante

Comment classer une page web en termes de popularité afin d'établir un classement qui permettra de hiérarchiser des réponses à apporter lors d'une recherche menée sur un moteur de recherche ?

Problématique



Voici un modèle (extrêmement) réduit du Web puisqu'il ne comporte que trois pages !

Les liens entre ces pages sont modélisés par un graphe.

On suppose qu'un 'surfeur aléatoire' est actif sur ce réseau.

À quelle fréquence ce surfeur se retrouvera-t-il sur chacune des pages P_1, P_2 et P_3 ?

Déroulement possible du projet

En amont du projet :

Les propositions suivantes permettent d'intégrer l'activité dans le cadre d'une réflexion à mener autour du thème des moteurs de recherche.

- Un groupe d'élèves peut réaliser un travail montrant que les résultats d'une requête ne sont pas tout à fait les mêmes selon le moteur de recherche utilisé.
- Un groupe d'élèves peut présenter une recherche d'une page par un moteur de recherche, en particulier le 'page rank'.
- Un groupe d'élèves peut présenter la notion de référencement.
- Un groupe d'élèves peut préparer un exposé sur le modèle du 'surfeur aléatoire'.

Différentes évolutions possibles pour le projet :

Les propositions suivantes donnent des pistes pour gérer les différences de vitesse d'exécution des élèves.
On peut aussi considérer ces différentes étapes comme une manière de séquencer le projet pour ceux qui en auraient besoin.

- 1ère étape : la première situation est mise en actes par des lancers de dés afin que tous les élèves comprennent le principe utilisé permettant de mettre en place un algorithme.
- 2ème étape : l'algorithme précédent est codé en PYTHON.
- 3ème étape : d'autres situations (d'autres graphes) sont proposées pour s'interroger sur diverses situations et adapter algorithmes et codes.

A la suite du projet :

Synthétiser ce travail par rapport au cours de SNT :

- Pourquoi est-il important de comprendre le principe de l'algorithme de recherche d'un moteur de recherche ?
- Pourquoi le référencement d'un page est-il important ?
- Comment est-il possible de rendre 'artificiellement' populaire une page et comment un moteur de recherche peut-il contourner ces pratiques ?

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Proposition de résolutions

Avec des dés :

- On propose aux élèves de 'jouer au surfeur aléatoire' à l'aide de dés quand cela est nécessaire (pour la première page et lorsque l'on tombe sur la page P_1).
- On collecte les résultats de l'ensemble des groupes (en étant critique sur les résultats donnés).
- On détermine la fréquence d'apparition de chaque page, ce qui donne la popularité de chacune d'elle.

Avec la calculatrice :

- Il s'agit à présent de coder cette situation en créant une fonction qui retournera la fréquence d'apparition de chaque page après **n** 'clics' du surfeur.

Etapes de résolution

L'importation de la bibliothèque *random* permet d'utiliser la fonction **randint**. On a choisi ici une syntaxe qui n'importe que la fonction **randint** parmi toutes les fonctions que contient cette bibliothèque.

La fonction **pagesuiv1** va permettre de suivre le graphe proposé précédemment:

- Si on se trouve sur la page 1, la page suivante est, au hasard, la page 2 ou la page 3 ;
- Si on se trouve sur la page 2, la page suivante est la page 1 ;
- Si on se trouve sur la page 3, la page suivante est la page 2.

Ces trois cas impliquent l'utilisation des conditions **if** / **elif** / **elif** qui sont accessibles par **Fns** / **Ctl**.

```
ÉDITEUR : PAGERANK
LIGNE DU SCRIPT 0002
from random import randint

#cas 1
def pagesuiv1(pa):
  if pa == 1:
    ps = randint(2,3)
  elif pa == 2:
    ps = 1
  elif pa == 3:
    ps = 2
  return ps
Fns… | a A # | Outils | Exéc | Script
```

On crée une fonction **rep** qui prend pour paramètre **n**, le nombre de répétitions de l'application du principe du 'surfeur aléatoire' ; elle renvoie la fréquence d'apparition de chacune des pages.

On utilise des compteurs stockés dans les variables **c1, c2 et c3** qui compteront le nombre de fois où l'on passe - respectivement - sur les pages P_1, P_2 et P_3. Ces compteurs sont initialisés à 0.
Sur la même ligne, on initialise la variable **pa** (qui représente le numéro de la page) à un nombre aléatoire entier compris entre 1 et 3.

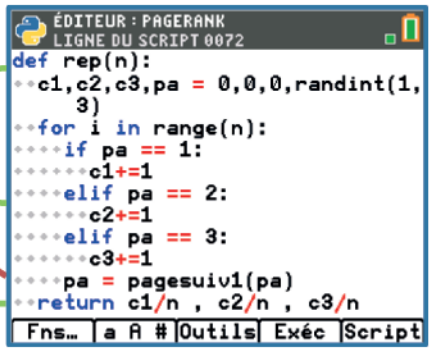
```
ÉDITEUR : PAGERANK
LIGNE DU SCRIPT 0072
def rep(n):
  c1,c2,c3,pa = 0,0,0,randint(1,
    3)
  for i in range(n):
    if pa == 1:
      c1+=1
    elif pa == 2:
      c2+=1
    elif pa == 3:
      c3+=1
    pa = pagesuiv1(pa)
  return c1/n , c2/n , c3/n
Fns… | a A # | Outils | Exéc | Script
```

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

La variable **pa** est égale au numéro de la page visitée par le surfer.
On répète **n** fois le fait d'incrémenter (augmenter de 1) le compteur de la page visitée par le surfer aléatoire.

Une fois sortie des trois conditions, la variable **pa** prend la valeur **pagesuiv1(pa)** c'est-à-dire que l'on suit la règle édictée par le graphe.
C'est l'utilisation successive de fonctions simples en python, en particulier le fait d'utiliser une fonction dans une fonction qui rend ce langage efficace.

On retourne la fréquence d'apparition de la page P_1, de la page P_2 et de la page P_3.



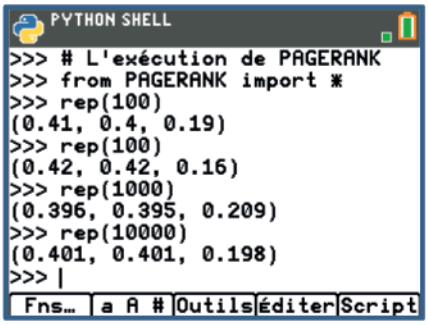
Rappel : `c+=1` est un raccourci pour `c=c+1`

Remarque

Une fois le programme fonctionnel, on peut formuler plusieurs remarques :

- Les résultats sont cohérents avec ce qui avait été collecté suite aux expérimentations avec les dés ;
- Les pages P_1 et P_2 ont une popularité similaire ;
- Plusieurs essais avec la même valeur de **n** ne donnent pas tout à fait les mêmes résultats ;
- Plus **n** est grand, plus on semble obtenir des résultats proches.

Ces deux derniers points peuvent éventuellement être réinvestis en cours de mathématiques.



Pour aller plus loin

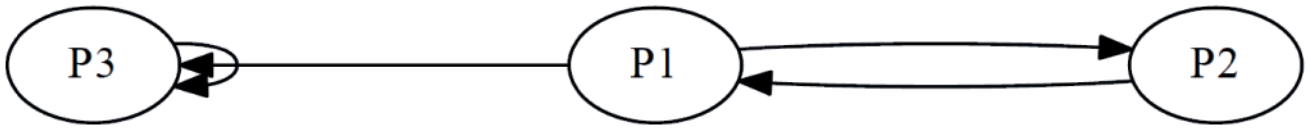
On peut alors proposer d'autres graphes modélisant les liens entre les pages et dans chaque cas, observer le problème posé et y apporter une solution.

Un nouveau graphe demande juste à créer une nouvelle fonction du type **pagesuiv** ; le cœur du programme restera le même.

Il faudra juste modifier dans la fonction **rep** l'appel à la fonction **pagesuiv** en la nommant en cohérence avec la situation étudiée.

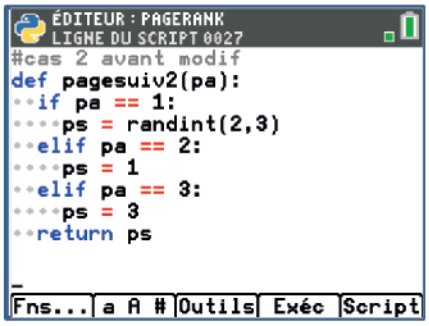
Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

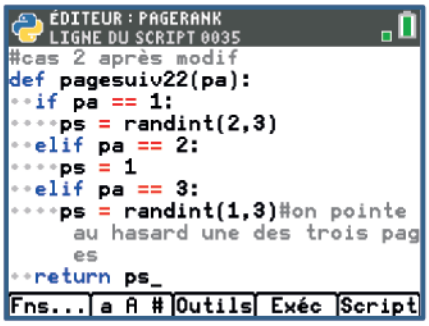
Ci-dessous, des propositions de graphe avec le problème posé et la solution proposée :



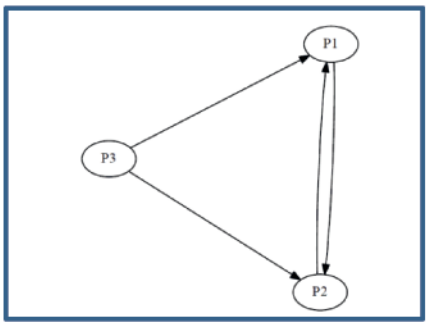
Dans ce cas-là, la page P_3 va 'absorber' les autres : une fois que l'on s'y trouve, on ne la quitte plus ! On parle de 'page puits'.

Proposition d'aménagement : lorsque l'on se trouve sur la page P_3, on pointe au hasard vers P_1, P_2 ou P_3. Cela donne la nouvelle fonction **pagesui22** :





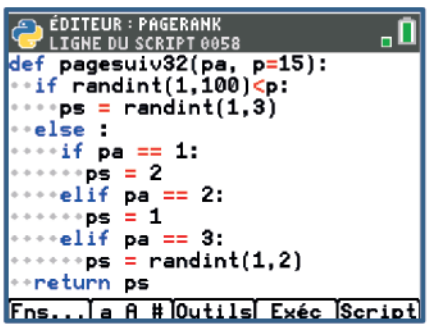
Dans ce troisième cas (présenté ci-contre), à moins de tomber au départ sur la page P_3, on n'y retourne jamais !



Pour pallier ce problème, on propose d'aller sur une des trois pages au hasard avec une probabilité p (de l'ordre de 0,15 en général) et le reste du temps, de suivre les indications du graphe.

Cela donne la fonction **pagesuiv32** ci-dessous, pour laquelle on a pris comme valeur par défaut de probabilité 0,15.

Par la suite, un appel de la fonction **rep** simulera n répétitions avec une probabilité d'aller sur une des trois pages au hasard de 0,15, et de suivre le cheminement du graphe avec une probabilité de 0,85.



Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Compétences visées

Un des objectifs de l'enseignement de SNT est de comprendre le principe de fonctionnement d'appareils de type GPS, afin de les envisager autrement qu'une simple "boîte noire" qui donne des résultats.

Le programme de SNT propose dans la partie Localisation, "*cartographie et mobilité*" de travailler sur "*les récepteurs GPS [qui] fournissent la localisation sous une forme normalisée facilement décodable, par exemple selon le protocole NMEA 0183 (National Marine Electronics Association), ou directement dans les métadonnées EXIF d'une photo.*"

On propose dans cette activité de construire plusieurs fonctions au sein d'un programme qui aura importé des données.

Ceci est à mettre en lien avec la partie "*Les données structurées et leur traitement*" avec un script qui a comme objectif de fournir des données et un autre qui réalisera le traitement de ces données.

Le script Python mis en jeu fait appel à des fonctions et des techniques en lien avec les chaînes de caractères.

Situation déclenchante

Les systèmes de localisation par satellites (américains, européens, chinois et russes) envoient des données codées selon divers protocoles.

Comment décoder ces données et pouvoir ainsi les interpréter ?

Problématique

Une trame NMEA 0183 nous étant donné, créer des scripts en Python permettant de :

- Déterminer le système de positionnement par satellite à l'origine de cette trame ;
- Déterminer l'heure correspondant à cette trame ;
- Déterminer la longitude d'une part, la latitude d'autre part du point correspondant.

Déroulement possible du projet

En amont du projet :

Les propositions suivantes permettent d'intégrer l'activité dans le cadre d'une réflexion à mener autour du thème de la localisation par satellites.

- Un groupe d'élèves peut réaliser un travail montrant le principe de géolocalisation par trois satellites et expliquer pourquoi en pratique il en faut quatre.
- Un groupe d'élèves peut présenter un historique du système de positionnement par satellite, en particulier du développement des systèmes GPS et Galileo.
- Un groupe d'élèves peut présenter les enjeux sous-jacents à la maitrise d'un système de positionnement par satellites.
- Un groupe d'élèves peut par ailleurs chercher des fonctions Python pour traiter des chaines de caractères.

Différentes évolutions possibles pour le projet :

Les propositions suivantes donnent des pistes pour gérer les différences de vitesse d'exécution des élèves.
On peut aussi considérer ces différentes étapes comme une manière de séquencer le projet pour ceux qui en auraient besoin.

- 1ère étape : L'enseignant donne un script à l'ensemble de la classe contenant plusieurs trames NMEA nommées t1, t2 …etc. La méthode la plus efficace sera un transfert de calculatrice à calculatrice du script contenant ces données. Les élèves créent tous un nouveau script (qui traitera ces données) qui commence par l'importation de ces données.
- 2ème étape : Une fonction permettant de retourner le type de système de positionnement par satellite est créée en classe ; l'ensemble des élèves peut chercher une fonction qui retournera l'heure à laquelle ce positionnement a été effectué.
- 3ème étape : D'autres fonctions peuvent être créées par les élèves plus à l'aise, donnant la longitude et la latitude. On peut aussi donner l'une des deux et demander aux élèves de trouver l'autre, ou alors de donner ces fonctions en partie et laisser les élèves les compléter.

A la suite du projet :

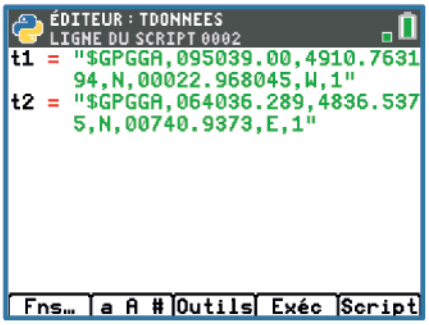
Synthétiser ce travail par rapport au cours de SNT :

- Est-il courant de séparer les données et leur traitement (des exemples) ?
- Existe-t-il d'autres types de trames liées au système de positionnement par satellites ?
- Quel est l'intérêt d'avoir un protocole tel que le protocole NMEA ? Par qui est-il fréquemment utilisé ?

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Proposition de résolutions

- L'enseignant (ou des élèves) peut présenter des techniques liées aux chaines de caractères, en particulier les manières d'extraire des éléments d'une chaîne de caractères par **[:]**.
- On donne aux élèves le script **TDONNEES** qui contient plusieurs trames saisies par l'enseignant (la saisie peut se faire sur ordinateur et le script peut être transféré sur calculatrice via le logiciel *TI Connect*).
- On remarquera que, pour le type de système de positionnement comme pour l'heure, les éléments sont toujours positionnés à la même place dans les trames NMEA, ce qui n'est pas le cas pour les valeurs de longitude et latitude (en effet, selon les valeurs numériques codées pour l'heure, les valeurs données pour longitude et latitude ne seront pas exactement à la même place dans la chaîne de caractères de la trame).
- Le traitement du type de système et de l'heure d'une part, et de la longitude et de la latitude d'autre part ne seront pas tout à fait du même ordre, demandant des compétences différentes. Cela peut permettre de différencier le travail au sein du groupe classe.



Etapes de résolution

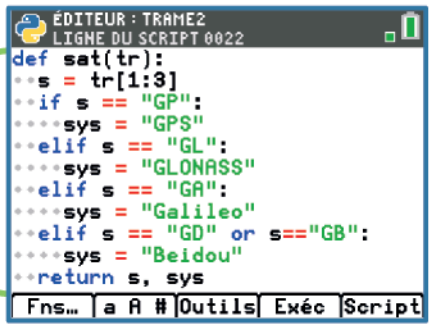
L'importation du script **TDONNEES** peut être facilité en allant chercher **from SCRIPT import *** dans le catalogue. Pour cela :

- Touche **2nde** suivie de la touche **0**
- Touche **alpha** suivie de **resol** pour obtenir la lettre F
- Descendre avec les flèches directionnelles.

Ne reste plus qu'à saisir lettre à lettre **TDONNEES**.

La fonction **sat** prend comme argument une chaine de caractères qui représente une trame ; on en extrait les caractères situés aux positions n°1 et n°2 de la chaîne (le premier élément de la chaîne est à la position 0) :

- Si ces caractères sont **GP**, on retourne comme système GPS ;
- Si ces caractères sont **GL**, on retourne comme système GLONASS ;
- Si ces caractères sont **GA**, on retourne comme système Galileo ;
- Si ces caractères sont **GD** ou **GB**, on retourne comme système Beidou.



Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

La fonction **h** prend comme paramètre une chaîne de caractères correspondant à une trame NMEA :

- les caractères situés en position 7 et 8 sont ceux de l'heure ;
- les caractères situés en position 9 et 10 sont ceux des minutes ;
- les caractères situés en position 11 et 12 sont ceux des secondes ;
- les autres caractères qui seraient éventuellement présents ne sont pas pris en compte ici.

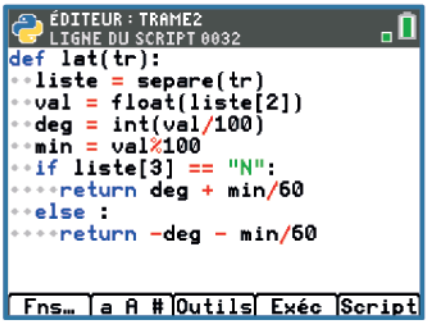
On retourne une chaîne de caractères comprenant ces données ainsi que les caractères " h ", " mn " et " s " afin de rendre le résultat lisible.
Les caractères sont concaténés à l'aide du symbole +

Pour aller plus loin

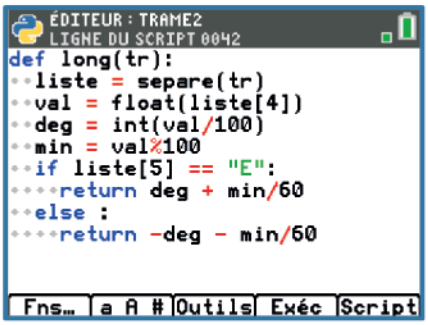
Pour travailler sur la longitude et la latitude, il va falloir distinguer les éléments de la trame NMEA séparés par des virgules.
Pour cela, on utilise la méthode **.split** (méthode native à python) qui s'applique à une chaîne de caractères, en indiquant comme paramètre l'élément séparateur.
Cette méthode renvoie une liste dont les éléments sont différentes chaînes de caractères.

Ainsi, **separe("$GPGGA,095039.00,4910.763194,N,00022.968045,W,1")** sera la liste ["($GPGGA", "095039.00", "4910.763194", "N", "00022.968045", "W", "1"] ; les éléments situés en position 2 et 3 permettront de déterminer la latitude et les éléments situés en position 4 et 5 permettront de déterminer la longitude.

- La partie utile de la trame NMEA pour déterminer la latitude est la chaine située en position n°2 : on la stocke dans la variable **val** ; **liste[2]** étant une chaîne de caractères, on force le type en indiquant **float** pour que cette chaîne devienne un nombre.
- Il s'agit ensuite de convertir des valeurs données en *degrés / minutes* en grandeur décimales. C'est l'objet des variables **deg** et **min**.
- Il faut ensuite tenir compte de la position Nord ou Sud indiquée dans la trame par **N** ou **S** ; selon le résultat, le résultat donné sera positif ou négatif, selon les conventions habituelles.



Le travail à réaliser pour déterminer la longitude est évidemment très similaire à celui sur la latitude.



Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

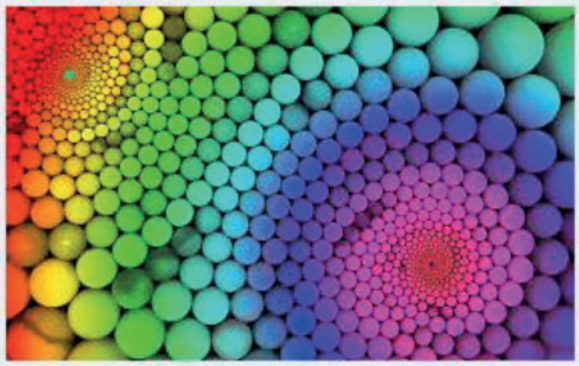
Image couleur /niveaux de gris

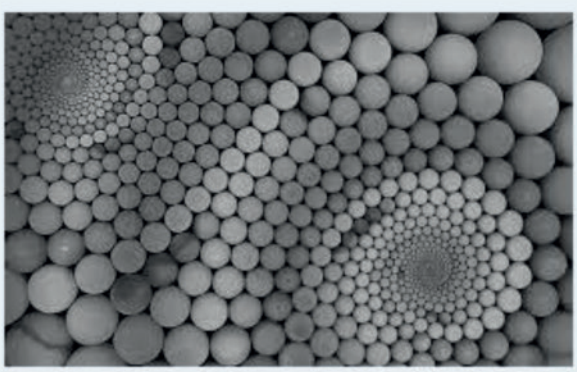
Compétences visées

Un des objectifs de l'enseignement de SNT est de développer et de coder des scripts PYTHON afin d'apporter une réponse à une problématique précise. A travers le thème " la photographie numérique", nous pouvons notamment travailler les compétences suivantes dans l'activité proposée :

- Traiter algorithmiquement une image pour la transformer en agissant sur les trois composantes de ses pixels.
- Écrire et développer des algorithmes pour résoudre une problématique.
- Connaître le système RGB et la structure d'une image numérique.

Situation déclenchante





Des algorithmes permettent aujourd'hui de traiter toutes les lumières, d'effectuer une retouche facile, avec une excellente qualité. Avec l'arrivée du téléphone mobile, des algorithmes de retouche d'images se multiplient notamment à travers les filtres. Comment ces algorithmes fonctionnent-ils ?

Problématique

Comment programmer un algorithme qui permette de transformer une image couleur en une image avec différents niveaux de gris ?

Fiche méthode

Déroulement possible du projet

Travail de groupe possible. En amont de ce projet, il serait intéressant que les élèves présentent plusieurs exposés :

- Un exposé sur la structure d'une image numérique.
- Un exposé sur le système RGB.
- Un exposé sur la méthode de conversion d'une image couleur en une image en niveaux de gris par moyenne des pixels RGB.

Ces exposés permettront d'acquérir les bases nécessaires pour la réalisation de ce projet.

Il faut ensuite choisir une image avec un nombre maximal de pixels de 320 × 190, parmi les fichiers exemples proposés (IMAGE1: 9 × 5 pixels ; IMAGE2 : 284 × 177 pixels ; GO : 221 × 190 pixels).

Proposition de résolution

Bibliothèque nécessaire (voir paragraphe importation de bibliothèque ci-après)

Les 2 paramètres de la fonction correspondent aux nombres de pixels (largeur et hauteur) de la photo utilisée.

L'instruction **plt.gr.drawImage("IMAGE2",0,50)** permet l'affichage de l'image nommée IMAGE2 avec le coin gauche de l'image au niveau du pixel (0,50) sur l'écran de la calculatrice. Pour rappel, l'écran de la calculatrice possède 320 × 240 pixels. Instruction à taper lettre par lettre (instruction absente des menus pour l'instant).

La double boucle *for* permet de passer en revue tous les pixels de l'image.

L'instruction **plt.gr.getPixel(x,y+50)** permet de récupérer le code RGB du pixel en position (x,y+50) sur l'écran de la calculatrice.

Instructions qui permettent la conversion en niveaux de gris.

L'instruction **plt.gr.setPixel(x,y+50,rbg)** permet l'affichage du pixel en position (x,y+50) sur l'écran de la calculatrice avec la couleur *rgb* obtenue.

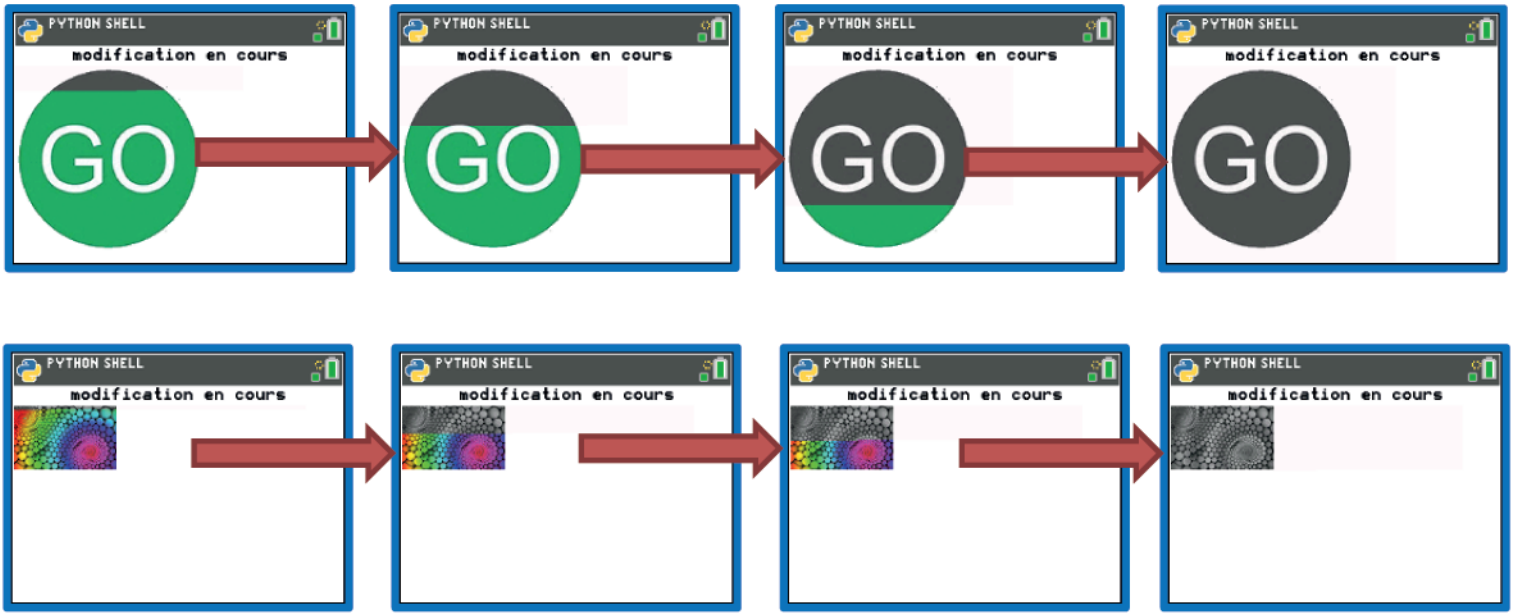
```
ÉDITEUR : GRIS
LIGNE DU SCRIPT 0001
import ti_plotlib as plt

def gris(largeur,hauteur):
  plt.cls()
  plt.title("modification en cou
    rs")
  plt.gr.drawImage("IMAGE2",0,50
    )
  for y in range (hauteur):
    for x in range (largeur):
      rgb = plt.gr.getPixel(x,y+
        50)
      R = rgb[0]
      G = rgb[1]
      B = rgb[2]
      s = (R + G + B)/3
      rgb = (s,s,s)
      plt.gr.setPixel(x,y+50,rgb
        )
  plt.show_plot()
  return
_
Fns… | a A # | Outils | Exéc | Script
```

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Fiche méthode

Voici des exemples de résultat du script : on observe la modification de l'image ligne par ligne de pixels (on visualise ainsi le fonctionnement de la double boucle *for*).



Le temps de traitement du programme peut parfois être assez long en fonction du nombre de pixels de l'image. Pour une exécution rapide on pourra utiliser le fichier IMAGE1 (9 × 5 pixels).

Utilisation des bibliothèques

Une fois le nouveau script créé, pour importer la bibliothèque *ti_plotlib*, appuyer sur la touche f(x), sélectionner la rubrique Modul, puis suivre les copies d'écran suivantes :

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Plus d'informations sur le site internet de T³ France :

- Des ressources pédagogiques pour votre classe
- Un programme de formations gratuites sur site et en ligne
- Des vidéos d'aide à la prise en main de la technologie

t3france.fr

Un service après-vente est également accessible depuis le site **education.ti.com/fr/csc**